Jim Mc Carthy

XÂY DỰNG GIẤC MƠ TRÊN VAI NGƯỜI KHỔNG LỒ
BẬT MÍ THÀNH CÔNG CỦA MICROSOFT

Chìa khóa trên đường hội nhập

BÍ QUYẾT THÀNH CÔNG CỦA TẬP ĐOÀN MICROSOFT

Microsoft Windows xp TM

VK

CHÌA KHOÁ TRÊN ĐƯỜNG HỘP NHẬP

Dịch từ tiếng Trung Quốc,
Nhà xuất bản Công nghiệp máy Trung Quốc

Xây dựng giấc mơ trên vai người khổng lồ

Bật mí thành công của tập đoàn Microsoft

CHÌA KHOÁ TRÊN ĐƯỜNG HỘP NHẬP

BÍ QUYẾT THÀNH CÔNG CỦA TẬP ĐOÀN MICROSOFT

Jim McCarthy

Người dịch: Mạnh Linh - Minh Đức

NHÀ XUẤT BẢN VĂN HOÁ - THÔNG TIN

LỜI NHÀ XUẤT BẢN

Nhà xuất bản Microsoft được Công ty Microsoft - đại gia trong lĩnh vực phần mềm - thành lập vào năm 1983. Từ khi thành lập đến nay, mỗi năm Nhà xuất bản Microsoft xuất bản khoảng 200 đầu sách mới và các sản phẩm đa phương tiện, từ những giáo trình học tập cho người dùng thông thường đến các loại sách tham khảo cho nhân viên chuyên nghiệp trong lĩnh vực công nghệ thông tin. Còn *cuốn sách này liên quan đến kinh doanh quản lý phần mềm, dựa trên các góc độ của ngành phần mềm, nội bộ Công ty Microsoft, và các khía cạnh kinh doanh quản lý để mô tả chi tiết phương thức làm việc và phong cách quản lý của tập đoàn Microsoft*. Điều này hoàn toàn khác với những cuốn sách trước mô tả Microsoft trên lập trường hoặc góc độ của người ngoài cuộc. Tác giả của cuốn sách là người có kinh nghiệm thực tiễn phong phú và thiết thực, đảm bảo bạn có thể thực sự hiểu rõ "bộ mặt thật" của Microsoft.

Microsoft được cả thế giới công nhận là đại gia phần mềm máy tính, các sản phẩm được sản xuất ra, dù là hệ điều hành, phần mềm ứng dụng, phần mềm mạng hay các thiết bị ngoại vi như bàn phím, chuột đều được người sử dụng ưa thích. Đằng sau vẻ bề ngoài này,

làm thế nào mà Microsoft chế tạo được các sản phẩm như vậy? Họ tìm hiểu nhu cầu người tiêu dùng như thế nào? Thiết kế sản phẩm theo yêu cầu khách hàng như thế nào? Làm thế nào để thiết kế sản phẩm theo đúng tiến độ? Nâng cao chức năng sản phẩm bằng cách nào? Làm thế nào để tạo ra kỹ thuật mới đảm bảo luôn dẫn đầu? Chúng ta đều biết rằng: Sự phát triển của kỹ thuật thông tin rất nhanh chóng, trong khi đó chu kỳ nâng cấp sản phẩm thông tin ngày càng ngắn lại. Làm thế nào để cạnh tranh trong lĩnh vực thông tin biến đổi "Chóng mặt" này, đều là chủ đề quan tâm của mỗi nhân viên ngành thông tin.

"*La Mã không phải được xây trong một ngày*", kinh nghiệm thành công của Microsoft là bài học rất có giá trị cho mỗi chúng ta!

Xin trân trọng giới thiệu cùng bạn đọc!

NXB VĂN HOÁ - THÔNG TIN

LỜI GIỚI THIỆU

Cuốn sách "Bí quyết thành công của tập đoàn Microsoft" do Jim McCarthy viết là cuốn sách gì vậy? Có thể so sánh nó như một cuốn sách liên quan đến phát triển phần mềm do Leonard de vonci (*Danh hoạ thời Phục Hưng, Italia - ND*), Tom Peter hợp sức viết.

Đây là cuốn sách chỉ nam phát triển phần mềm có tầm nhìn xa, sinh động và thực tế, hướng dẫn cách phát triển phần mềm tốt và hoàn thành đúng tiến độ. Nói đúng hơn, Jim Carthy hướng dẫn bạn cách xây dựng và dẫn dắt nhóm phát triển phần mềm xuất sắc.

Tôi và Jim McCarthy là cộng sự tốt khi cùng làm việc cho Microsoft, chúng tôi cùng tham gia thiết kế Visual C++. Theo tiêu chuẩn Microsoft, phần mềm này tạm được; còn dưới góc độ của người sử dụng thì đây là một kiệt tác. Từ khi chính thức phát hành đến nay, lượng tiêu thụ đã lên tới hàng triệu bộ, được đánh giá rất khả quan.

Nhưng cuốn sách này không bàn về Visual C++ hay Microsoft, mà nói đến những kinh nghiệm quý báu tích luỹ được trong quá trình phát triển Visual C++ tại Microsoft.

Ngay cả trong nội bộ phát triển phần mềm chuyên nghiệp của Công ty Microsoft, nhóm phát triển Visual C++ của chúng tôi được công nhận là ưu tú: Hoàn thành đúng thời hạn, thường xuyên vượt tiêu chuẩn. Đa số mọi người đều quen với việc chậm trễ trong phát triển phần mềm, trong khi đó chúng tôi lại bán sớm hơn cả dự kiến, đây là việc khó thực hiện trong vài năm trước, và đến tận bây giờ nhiều nhóm phát triển phần mềm vẫn không làm nổi. Vài năm trước, cứ một đến hai năm chúng tôi lại đưa ra một phiên bản mới, còn hiện giờ thì cứ 4 tháng một phiên bản.

Nếu công việc hay kinh nghiệm của bạn có liên quan đến phát triển phần mềm, tôi dám đảm bảo rằng bạn sẽ tìm thấy đáp án trong cuốn sách này. Khả năng quan sát nhạy bén, thông minh của Jim McCarthy và cách hướng dẫn thực tế sẽ giúp bạn thấy những gì ông ta kể giống như câu chuyện của chính bạn vậy.

Cuốn sách này là đúc kết kinh nghiệm trong thực tế của tác giả, được thực hiện trong học viện phần mềm: "Chúng ta không được phép thất bại".

Nội dung của cuốn sách được viết trên cơ sở bài phát biểu "*12 bí quyết đưa ra sản phẩm xuất sắc*" của Jim McCarthy. Bài phát biểu đã thu hút hàng ngàn công chúng, tiếng vỗ tay không ngớt.

Phương thức phát triển phần mềm của Jim McCarthy ngày càng trở nên thành thục, đồng thời luôn phù hợp với yêu cầu thực tế của các doanh nghiệp. Các doanh nghiệp cần những thứ phức tạp mà đa dạng, luôn tìm kiếm những cái mới, tuy gặp nhiều thách thức, nhưng cũng rất thú vị. Qua cuốn sách bạn cũng sẽ thấy được tính cách đặc sắc của Jim McCarthy - *luôn đi trước bản thân*.

Lần đầu tôi gặp Jim McCarthy là vào năm 1992, lúc đó ông chỉ có một chiêu (hoặc chỉ cho tôi biết một chiêu) là: "*Đừng vì vấp ngã mà không đứng dậy*". Khi đó, tôi vừa mới thành lập ban ngôn ngữ hệ thống của Microsoft (nay đã bỏ), mời Jim McCarthy đến phỏng vấn, ông ta đã để lại cho tôi một ấn tượng khá sâu sắc. Jim McCarthy đến từ công ty Whitewater, đã tham gia phát triển ngôn ngữ lập trình hướng đối tượng Actor. Hôm đó ông mặc quần bò, và tôi cũng thế, có lẽ đó là do chúng tôi có duyên với nhau.

Đương nhiên tôi chọn cộng sự không phải do ăn mặc giống tôi, mà là tuyệt chiêu của Jim McCarthy. Nghĩ lại, nguyên nhân chính mà Jim McCarthy đã thu hút sự chú ý của tôi là ông đã rút trong túi ra một quyển sổ, phía trên ghi rõ mười lý do chính phải tuyển ông ta. Jim McCarthy đúng là một người rất đặc biệt!

Jim McCarthy là như thế, ông không ngừng trưởng thành: Ông ghi lại những kinh nghiệm, những quan sát thấy được cũng như những bí quyết thành công. Mấy năm gần đây, sổ ghi chép của ông đã phong phú đến mức có thể làm thành một quyển sách đặc sắc của Nhà xuất bản Microsoft, đồng thời là kiệt tác trong lĩnh vực phát triển phần mềm.

Mấy năm qua tôi cùng Jim McCarthy làm việc, cùng chia sẻ kinh nghiệm cho nhau, nay tôi mong rằng cuốn sách này sẽ sẻ chia kinh nghiệm cùng bạn.

Dennis Gibbert

GIỚI THIỆU TÁC GIẢ

Jim McCarthy tham gia phát triển phần mềm máy tính đã được 25 năm.

Bắt đầu làm phần mềm từ năm 1976, sử dụng ngôn ngữ BASIC phiên bản đầu tiên trên máy TRS-80 Model I; Từ đó ông rời bỏ tất cả mọi hứng thú khác để chú tâm vào phát triển phần mềm máy tính. Sau vài năm kinh nghiệm với Trash 80 và BASIC, Jim McCarthy mang theo bộ biên dịch BDS C và vài quyển sách, lái xe đến ẩn cư vài tuần trong rừng, làm việc miệt mài không nghỉ nhằm tìm ra "Ngôn ngữ lập trình thực sự".

Thử nghĩ xem, ánh sáng của màn hình máy tính với tiếng kêu ro ro của ổ cứng, của quạt, cộng với tiếng kêu râm ran của ếch nhái, thật là một bức tranh yên tĩnh và đặc biệt.

Sau đó Jim McCarthy đã thành lập một công ty tư nhân, không lâu sau, ông đến làm tại phòng thí nghiệm của AT&T, công ty Whitewater (Nước trắng) và công ty Microsoft. Hiện nay ông là Giám đốc phòng hành chính sự phiệp của Visual C++ thuộc Microsoft. Mặc dù ông hay tham gia phát biểu, thảo luận tại các diễn đàn trên thế giới, nhưng phát triển phần mềm vẫn là trọng tâm

công tác của ông, vì đây là công việc được ông yêu thích nhất.

Ngoài ra Jim McCarthy còn vẽ tranh, sáng tác, khắc đá hoặc gỗ, đồng thời còn là một ông bố đầy trách nhiệm. Đôi khi trong khu rừng, bạn vẫn thấy lấp lánh ánh sáng màn hình máy tính của Jim McCarthy.

LỜI TÁC GIẢ

Cuốn sách này được viết dựa vào những kinh nghiệm có được trong suốt quá trình phát triển phần mềm đầy những biến đổi. Quá trình phát triển phần mềm sở dĩ vất vả nhưng đầy hấp dẫn là do nó chứa đựng đầy tính bất định, cả quá trình là một động thái luôn biến đổi, bạn phải thuận theo xu thế, chuyển hoá tính động này thành một sức sáng tạo phong phú, chứ không được để nó "xỏ mũi" dắt bạn đi. Đây chính là chủ đề thảo luận chính của cuốn sách, đưa ra những giải đáp về vấn đề phát triển phần mềm dựa trên sự tâm đắc trong quá trình phát triển Visual C++ của Microsoft.

Viết kinh nghiệm của mình thành sách, có thể có một vấn đề nhỏ cần được làm rõ: đó là tôi đã lấy quan điểm và cảm nhận của mình để kể những câu chuyện này, nhưng những người tham dự có thể có cách nhìn nhận khác, những điều được tôi nhấn mạnh đều là những điểm tôi hứng thú hoặc cảm thấy là quan trọng. Khi tôi kể những nội dung này, thường nhấn vào các trọng điểm khác tuỳ theo yêu cầu của người nghe, nhưng khi viết thành sách thì không thể làm như vậy.

Trên thực tế, kiến giải cùng một vấn đề, cùng với kinh nghiệm tích luỹ ngày càng phong phú có thể càng

sâu sắc chi tiết hơn. Tôi mất một năm rưỡi dùng những nghỉ ngày cuối tuần và các buổi tối mới viết xong cuốn sách này, trong thời gian đó, công việc phát triển phần mềm ban ngày vẫn không ngừng giúp tôi học thêm được nhiều điều mới mẻ. Tôi khẳng định chắc chắn rằng nội dung cuốn sách cần thiết cho bất cứ ai, bất cứ người nào làm công tác phát triển phần mềm, càng có nhiều kiến thức mới càng có thể làm việc tốt hơn, nghiên cứu càng sâu sắc hơn. Những năm tháng phát triển phần mềm cũng giống như tuổi đời của giống chó, người sống một năm tương đương chó sống bảy năm, một năm của con người tương đương N năm phát triển phần mềm, N chính là số cột mốc trong một năm. Quá trình viết cuốn sách này, vừa là N năm phát triển phần mềm, vừa có rất nhiều phần mềm ra đời. Thời gian trôi đi thật nhanh, mà phát triển phần mềm dường như làm cho thời gian càng trôi đi nhanh hơn, càng khó nắm bắt hơn.

Tôi đã nói chuyện với nhiều chuyên viên và giám đốc các công ty phát triển phần mềm, những ví dụ nêu trong sách đều là những kinh nghiệm khá giống với các chuyên viên làm phần mềm, dù đó là loại phần mềm gì. Mong rằng những quy tắc, quan điểm và kiến nghị mà tôi nêu ra, đã bao quát hầu như tất cả những vấn đề có thể gặp phải khi phát triển phần mềm, và có thể phát triển được bộ quy tắc phát triển phần mềm. Có được

tiêu chuẩn này - quan điểm chung của mọi người - công việc của chúng ta càng dễ thành công, đồng thời cố gắng phòng tránh những vấn đề chưa xảy ra. Do hiện nay kinh nghiệm tích luỹ được còn rời rạc, mọi người đều không thực sự xác định được cần làm những gì, thậm chí nhiều nhóm phát triển phần mềm vẫn không rõ cần làm gì để kiện toàn nhóm nên bị rơi vào trạng thái tâm lý bất ổn, khó có thể giải thoát.

Cuốn sách mô tả một nhóm người hoàn thành thành công một phần mềm trong thời gian nhất định, và sau đó không ngừng nâng cấp phiên bản mới đúng hẹn, luôn được thị trường chấp nhận. Trong thời gian phát triển phần mềm này, tôi đã tích luỹ - tổng hợp - chỉnh lý các kinh nghiệm, đồng thời đi sâu giải thích các sự kiện phát sinh, phân tích kỹ các vấn đề, phát triển thành 54 quy tắc. Tôi tin tưởng rằng những quy tắc này là bí quyết thành công của chúng tôi, và tin rằng sau một thời gian nữa đọc lại cuốn sách này, tôi càng có nhiều biện pháp sâu sắc hơn, cũng như phát hiện ra càng nhiều chỗ cần nhấn mạnh hơn trong cuốn sách này.

Cần chú rằng về bản chất cuốn sách là những ghi chép về các sự kiện và những điều tâm đắc, ghi chép những sự kiện đáng suy nghĩ trong phát triển phần mềm, ghi chép cách giải quyết vấn đề của một nhóm

xuất sắc, đồng thời cũng phân tích hành vi tổ chức của ngành khoa học kỹ thuật.

Hy vọng rằng cuốn sách này mới chỉ là mở đầu, các phiên bản mới tiếp sau sẽ càng thêm phong phú. Đồng thời mong rằng cuốn sách này sẽ có ích cho mỗi nhân viên làm công tác phát triển phần mềm.

Cảm ơn các cộng sự trong nhóm phát triển Visual C++ của Microsoft, nhiều biện pháp trong cuốn sách này thực chất là do thảo luận với họ mà có, nhất là Jeff Beehler, Brad Christian, Greg DeMichillie, ... Còn nhiều người khác không thể liệt kê hết, nhưng tôi phải đặc biệt cảm ơn Dave Moore và Chris Williams.

KHÁI QUÁT

BỐN THỜI KỲ PHÁT TRIỂN PHẦN MỀM

Trên lý thuyết, phát triển phần mềm thành công hầu như không có gì khó khăn cả: trước hết cần tìm hiểu nhu cầu khách hàng và định vị góc độ trên thị trường, sau đó thiết kế sơ đồ của sản phẩm phần mềm - nó càng phải phù hợp với nhu cầu khách hàng hơn các sản phẩm khác. Sau đó xây dựng nhóm phát triển để chế tạo sản phẩm, sau khi hoàn thành sẽ tung ra thị trường. Tiếp theo áp dụng các biện pháp kinh doanh để thông báo cho khách hàng tiềm năng biết về giá trị đặc sắc của sản phẩm, khách hàng sẽ vui mừng khi mua được sản phẩm, lúc sử dụng cũng rất hài lòng. Lúc đó công ty bạn sẽ nổi tiếng trong ngành phần mềm, được đưa lên bìa các tạp chí Fortune, Success,... trở thành đề tài cho mọi người ca ngợi.

Trên lý thuyết, hoàn thành sản phẩm đúng hạn rất đơn giản, nhưng trong thực tế, hầu hết các công ty đều thất bại.

Nghe thì cũng được, nhưng sự thực lại khác xa, cũng có thể là do chưa có bộ sách thực sự có ích về phát triển phần mềm. Một khi có mô hình phát triển phần mềm thành công, mọi người có thể kỳ vọng tiến hành theo các bước này để đạt được thành công. Phía sau chúng tôi sẽ nói rõ cách xây dựng và gắn kết kỳ vọng cho cả tổ chức, nhưng bạn cũng cần hiểu rằng, hoàn thành phần mềm thành công đúng hạn là một trong những công việc khó khăn nhất, mặc dù ai cũng hy vọng thực hiện được. Bạn không thể dựa dẫm vào các

công thức, mỗi phần mềm đều có đặc tính riêng, bất kỳ lúc nào phát triển phần mềm đều cần đặc biệt cố gắng mới được. Nhưng ngược lại, thành công trong phát triển phần mềm là điều vui sướng nhất: cần phải hiểu rõ nhu cầu khách hàng, cần phải xây dựng nhóm phát triển với hiệu quả cao, định nghĩa sản phẩm phần mềm cần dựa trên lợi ích kinh tế thị trường, đồng thời chương trình phần mềm phải chính xác không có lỗi. Sau đó, khi sản phẩm lớn đã ra đời, gây tiếng vang trên thị trường, các phương tiện truyền thông không ngừng giới thiệu sản phẩm của bạn, khách hàng mang tiền đến tranh nhau mua.

Đúng vậy, điều này hoàn toàn có thể xảy ra, cuốn sách này cho bạn biết bí quyết phát triển phần mềm thành công, những bí quyết này được viết lần lượt giúp người đọc dễ nhớ, đồng thời cũng thiết thực - súc tích nhưng có tác dụng lớn - có ảnh hưởng rất lớn đến định nghĩa, phát triển và kinh doanh sản phẩm phần mềm. Hoàn thành sản phẩm phần mềm tốt lại đúng hạn là rất khó, nhưng không có nghĩa là không làm được. Cuốn sách này có thể làm thay đổi quan điểm của bạn, những nhân tố trước đây bạn cho là quan trọng có thể không đáng để bạn quan tâm. Phương pháp và trình tự phát triển, năng lực kỹ thuật lập trình tốt và công cụ phát triển, cũng như cách quản lý đều có ảnh hưởng rất lớn đến thành công hay thất bại trong phát triển phần

mềm, nhưng việc những nhà quản lý có thể thực hiện được các quan điểm giới thiệu trong sách hay không mới là nhân tố có tính quyết định.

Phát triển phần mềm là công việc rất phong phú, và cuốn sách nhấn mạnh đến tính chỉnh thể của các nguyên tắc. Tôi cho rằng, nếu phát triển phần mềm là một hạng mục có thể làm theo tiến độ hoặc chức năng nhất định, không bằng nói đó thuộc loại giác quan thứ sáu. Tôi không coi nhẹ tầm quan trọng của tổ chức hay sự hợp tác mà muốn chỉ ra các hoạt động tâm lý phức tạp ẩn trong một tổ chức đang phát triển, bao gồm ý tưởng sáng tạo, năng lực, kỹ thuật... đây chính là nguồn cảm hứng sáng tạo, là năng lượng của nhóm phát triển. Nếu biết đưa vào sản phẩm phần mềm một cách thích hợp, sẽ làm cho phần mềm càng trở nên đặc sắc, khác hẳn những phần mềm hiện có.

Trên bề mặt tưởng chừng như yên tĩnh, thực chất phát triển phần mềm đầy biến động.

Ý TƯỞNG PHÁT TRIỂN PHẦN MỀM

Phần mềm là sản phẩm trí tuệ, về cơ bản phát triển phần mềm là một loại đầu tư trí tuệ, mỗi *bit* thông tin trên CD hay đĩa cứng đều là kết tinh trí tuệ của nhóm lập trình. Trí tuệ chứa trong phần mềm càng

lớn, giá trị phần mềm càng cao, càng được thị trường chấp nhận; phần mềm tốt sẽ không cần nhiều chi phí cho quảng cáo và tiếp thị, bản thân đã thu hút được rất nhiều khách hàng, đem lại lợi ích cho người sáng tạo ra nó cũng như người sử dụng.

Giám đốc hay trưởng nhóm phát triển phần mềm cần có một quan niệm đúng đắn, đó là chuyển trọng tâm của hoạt động, phát triển phần mềm vào quá trình đưa trí tuệ từ người vào máy tính. Đa số trưởng nhóm phát triển phần mềm đều không có nhận biết đúng đắn, cho rằng công việc của họ đại loại là thiết kế, kiểm tra phần mềm, xây dựng hệ thống tệp tin, tiếp thị sản phẩm, ... nên coi trọng việc quản lý quá trình phát triển phần mềm.

Việc hiểu nhầm này là nguyên nhân chính dẫn đến không làm đúng tiến độ. Dù chất lượng phần mềm tốt hay dở thì nó vẫn không ra mắt đúng hạn, việc để lạc hậu này dần dần trở thành thói quen. Loại phần mềm lạc hậu vì không đúng tiến độ này thường được gọi là "phần mềm bong bóng" bởi chúng có thể đi vào quên lãng bất cứ lúc nào.

Thực tế nhiệm vụ quản lý phát triển phần mềm là phát huy đầy đủ và thích hợp trí tuệ của thành viên trong nhóm. Trí tuệ này là trí óc trừu tượng của con người như sức sáng tạo, trí thông minh, hiệu suất, ...

Trọng tâm của sản phẩm trí tuệ là bạn phải dùng phương thức thông minh để kết hợp trí tuệ của một nhóm người, đây cũng là phần khó khăn nhất trong quá trình phát triển phần mềm.

Muốn hoàn thành sản phẩm lớn đúng hạn, cần phải có sự toàn tâm toàn ý của mỗi người.

Giả sử hỗ trợ về tài chính không là vấn đề quan trọng, mà điều cần chú ý là sự dốc sức của cả nhóm, sự kết hợp trí tuệ hữu hiện giữa các thành viên. Các quan niệm và nguyên tắc cơ bản trong sách đều rất dễ hiểu, chỉ là những kiến thức thông thường. Muốn tổng hợp trí tuệ của nhóm, cần phải thực sự làm được ba việc sau: Khởi phát suy nghĩ của nhóm, nắm hướng suy nghĩ đúng đắn, suy nghĩ có hiệu quả. Nếu bạn đã tham gia những nhóm trí tuệ như vậy, có thể bạn đã phát hiện ra ba bí quyết quan trọng này.

Vậy rốt cuộc vấn đề là ở chỗ nào? Nguyên tắc quản lý phát triển phần mềm đơn giản như vậy, thì tại sao vẫn có rất ít trường hợp thành công.

Trước hết chúng ta hãy xem qua mô hình vận dụng tài nguyên con người ở hầu hết các doanh nghiệp. Thông thường, tài nguyên con người ở các doanh nghiệp được đầu tư vào hai lĩnh vực: Một là hoạt động trí tuệ

có tính sáng tạo thuộc loại thiết kế hoặc kế hoạch; Hai là hoạt động máy móc có tính lặp lại, thuộc loại sản xuất hoặc thi công. Cho đến nay, đa số các doanh nghiệp đều dùng tài nguyên con người vào hoạt động máy móc có tính lặp lại. Tuy một cỗ xe, một toà nhà hay một con đường đều có sự đầu tư trí tuệ, nhưng so với toàn bộ tài nguyên con người thì vẫn chiếm tỷ lệ thấp.

Vậy chúng ta hãy xem, làm thế nào để tổ chức một doanh nghiệp với trọng tâm là hoạt động máy móc có tính lặp lại. "Hiệu quả là trên hết", đúng vậy, tất cả các giá trị gia tăng đều bắt nguồn từ đây. Henry Ford đưa ra quan niệm "*dây chuyền sản xuất*", mở ra một lịch sử mới trong phát triển công nghiệp thế kỷ 20, từ đó tất cả các hoạt động sản xuất đều như một cỗ máy lớn không tình cảm, rập khuôn, không ngừng lặp đi lặp lại: nguyên liệu không ngừng đưa vào từ một đầu, sản phẩm từ đầu kia không ngừng đi ra. Để quản lý cỗ máy lớn này, tổ chức quản lý phân lớp đã ra đời. Mỗi người trong tổ chức này chỉ như một con ốc nhỏ, nội dung công việc bị bó hẹp trong một phạm vi đã được xác định. Chỉ cần mỗi thành viên nỗ lực trong cương vị bé nhỏ của mình thì hiệu qủa tổng thể sẽ đạt được mức cao nhất.

Nhưng sản xuất phần mềm hoàn toàn không như vậy, vì đó là sản phẩm trí tuệ vô hình chứ không phải

thương phẩm hữu hình, hoạt động máy móc có tính lặp lại chỉ chiếm tỷ lệ rất nhỏ, còn hoạt động máy móc có tính sáng tạo mới có giá trị quyết định.

Việc đóng gói phần mềm tại xưởng sao chép không đơn giản, nhưng cũng không khó hơn các sản phẩm tiêu dùng thông thường khác (ví dụ máy chụp ảnh). Không nghi ngờ gì nữa, thách thức lớn nhất của công ty phần mềm là sản phẩm hoàn chỉnh đầu tiên, quá trình sáng tạo phần mềm rất phức tạp và đầy thay đổi, chỉ cần làm ra được mô hình là các việc khác sẽ trở nên đơn giản.

Tính phức tạp và thay đổi trong sản xuất phần mềm xuất hiện do bản chất hoạt động trí tuệ có tính sáng tạo. Đa số các tổ chức xí nghiệp không phải được thành lập để khuyến khích tư duy, do đó bạn phải thay đổi mô hình tổ chức, còn được gọi là tái tạo doanh nghiệp mới có thể xây dựng được môi trường phát triển phần mềm thích hợp.

Trong cuốn sách này, bạn sẽ nắm được hình thái tâm lý tổng thể của một công ty phần mềm thành công - Microsoft. Đối với những giám đốc muốn nắm được lịch trình phát triển phần mềm, điều này rất quan trọng. Đối với lập trình viên, việc hiểu được ý nghĩa then chốt của lịch trình cũng quan trọng không kém. Đồng thời, nếu mỗi thành viên nhóm lập trình không

hiểu được tính toàn cục của kế hoạch phát triển phần mềm thì cống hiến của họ chỉ hạn chế trong công việc được giao, và trở thành "máy" sản xuất phần mềm. Đó chính là sự lãng phí tài nguyên nhân lực và trí tuệ. Nếu bạn muốn phần mềm hoàn thành đúng hạn, cần phải làm cho mỗi thành viên dồn tâm trí vào việc này, và duy trì cho đến khi hoàn thành sản phẩm. Đây là nhiệm vụ quan trọng nhất của giám đốc phát triển phần mềm, cũng là như đề chính của cuốn sách.

BỐN THỜI KỲ PHÁT TRIỂN PHẦN MỀM.

Quá trình phát triển phần mềm chia thành bốn giai đoạn:

Giai đoạn bố cục (Opening Moves) bao gồm đưa ra chiến lược thị trường hợp lý, thiết kế sản phẩm, kế hoạch lập trình và tất cả các công việc mở đầu khác. Thách thức lớn nhất của giai đoạn này là xây dựng nhóm lập trình trong khi mọi thứ đang bề bộn ngổn ngang. Cũng giống như nước cờ đầu tiên trên bàn cờ, tạo cơ sở cho cả một quá trình. Tôi gọi ngày đầu tiên làm công việc này là "ngày hành động" (Moving day), đây cũng chính là cao trào của giai đoạn bố cục, các hạng mục được bắt đầu.

Giai đoạn giữa (Middle Game) chiếm hầu hết thời gian trong cả quá trình, quyết định sự thành bại

của phần mềm. Giai đoạn giữa có thể so sánh như Thánh kinh đã mô tả, trong 40 năm trường gian khổ, vô số nô lệ và giáo dân có tín ngưỡng kiên định, dưới sự chỉ đạo của Yahveh (Giê-hô-va) đã xây dựng nên một vương quốc xinh đẹp trên sa mạc hoang vu. Hạng mục phát triển phần mềm thường không cần tới 40 năm, nhưng những khó khăn và công sức bỏ ra, làm cho người ta cảm thấy thật sự phải lâu như vậy. Một nhóm người kiên trì đến đích sẽ giống như một thể sống thống nhất đầy kỳ diệu, ngoài tâm niệm và cống hiến kiên định ra, chỉ cần sớm hoàn thành sản phẩm, thoát khỏi giai đoạn "ác mộng" này là được.

Giai đoạn hoàn thành (Ship Mode) là cánh cửa thoát khỏi biển khổ. Cũng như đứa trẻ ra đời vậy, đầy đau đớn và lo lắng, nhưng không thể tránh khỏi. Cần phải qua giai đoạn này, phần mềm mới có mặt trên thị trường. Điều làm mọi người phấn chấn nhất là niềm vui khi sản phẩm hoàn thiện, là cao trào của giai đoạn hoàn thành, cũng là nỗi đau cuối cùng phải trải qua.

Giai đoạn công bố (Lauch) là thời khắc của không khí kinh doanh, phần mềm được công bố ra đời, bạn phải lựa được ấn tượng sâu sắc và khó phai trong tâm trí người sử dụng, tối thiểu là vẫn nhớ đến bạn khi phiên bản mới được phát hành sau đó. Qua giai đoạn này, phần mềm mới được coi như giao đến tận tay người sử dụng.

NẾU BẠN KHÔNG LÀM TRONG LĨNH VỰC PHẦN MỀM.

Có thể bạn không làm việc trong lĩnh vực phần mềm, cuốn sách vẫn rất có giá trị tham khảo. Tuy trọng tâm về phát triển phần mềm, nhưng nội dung đều là kinh nghiệm thực tiễn của tác giả, trong đó các nguyên tắc nguyên lý cơ bản nhất có thể thích hợp tất cả các lĩnh vực khác, tin rằng bạn sẽ dễ dàng hình dung được nội dung của cuốn sách giống với môi trường công việc mà bạn đang gặp phải.

PART One

PHẦN 1
GIAI ĐOẠN BỐ CỤC

Rất nhiều sách bàn về phát triển phần mềm đều giả sử bạn đang ở trong một tình trạng lý tưởng: nhóm phát triển rất chú tâm và dốc hết sức hoàn thành công việc, đồng thời hiểu rõ bản chất của nhiệm vụ; họ thu thập được các thông tin hoàn chỉnh, xây dựng quy cách thiết kế, và không ngừng cải tiến sản phẩm; họ mời người sử dụng tham gia, người sử dụng sẽ hợp tác tốt, và cùng nhóm phát triển phân tích nhu cầu sử dụng. Mọi việc đều được hình dung rất thuận lợi. Nhưng đáng tiếc là trạng thái lý tưởng đó không có trong thực tế, trong hạng mục phát triển phần mềm có thể bạn sẽ không nhìn thấy bóng dáng của những cảnh tượng thuận lợi mà chỉ là hàng loạt các vấn đề đau đầu.

Không chỉ có vấn đề mà còn có những thách thức đáng sợ. Những công việc của nhóm phát triển phần mềm xuất sắc làm trong giai đoạn bố cục này rất nhiều, có thể phân chia thành 5 phạm trù: tổ chức, cạnh tranh, khách hàng, thiết kế, phát triển. Công việc trong giai đoạn này rất đa dạng, đồng thời trong mỗi chi tiết đều có thể tổng hợp, bao hàm các kết quả mong muốn.

TỔ CHỨC NHÓM PHÁT TRIỂN

Tổ chức, là việc kết hợp nhóm người thích hợp đảm nhiệm các công việc sau và tham gia thiết kế:

Quản trị dự án (Program Management) phụ trách xây dựng lịch trình phát triển, liên hệ đối ngoại, tìm kiếm hỗ trợ kỹ thuật.

Bảo đảm chất lượng (Quality Assurance) kiểm tra và đánh giá chất lượng phần mềm.

Phát triển chương trình (Development) viết chương trình, tìm lỗi.

Quản lý sản phẩm/Marketing (Product Management/Marketing) phụ trách định vị sản phẩm, truyền thông tin chính xác cho khách hàng, kể cả công bố sản phẩm ra đời, liên hệ với giới truyền thông.

Văn bản hệ thống và đào tạo người sử dụng (Documentation/User education) đảm nhiệm việc sử dụng từ ngữ chính xác để biểu đạt hướng dẫn sử dụng sản phẩm.

Chú ý: Thiết kế ở đây là chỉ thiết kế sản phẩm phần mềm, chứ không phải đơn thuần là thiết kế chương trình mà mọi người gọi là "lập trình viên". Thực ra từ này được dịch từ programmer, tuy cũng tương đương với "thiết kế", nhưng công việc chủ yếu có tính

thi công, thực thi chứ không mang tính nghệ thuật. Thiết kế (phát triển) sản phẩm phần mềm rộng hơn, bao gồm thiết kế mục tiêu, định nghĩa các chức năng của sản phẩm, đáp ứng các nhu cầu gì, khách hàng mục tiêu là ai? cần phần cứng và hệ điều hành nào?

Bạn không nhất thiết phải chia nhóm phát triển thành năm bộ phận (mặc dù tôi thấy đây là cách làm đem lại hiệu suất cao nhất), nhưng những công việc này phải chắc chắn có người phụ trách. Điều cần chú ý là bộ phận nào cũng tham gia vào thiết kế, do đó mỗi thành viên đều hiểu rõ ràng và chỉnh thể về hạng mục đang làm, làm đồng nhất mục tiêu của mỗi thành viên.

Nếu nhóm phát triển không thể hợp tác, luôn có ý kiến bất đồng về mục tiêu đề ra thì công việc đầu tiên phải làm là tìm ra nguyên nhân chính gây mất đoàn kết.

NHÂN VIÊN QUẢN LÝ CHẤT LƯỢNG CÓ THUỘC NHÓM THIỂU SỐ?

Nếu nhân viên bảo vệ chất lượng cho rằng công việc của họ là kiểm tra chương trình, còn nhân viên phát triển cho rằng công việc của họ là viết chương trình cho nhân viên bảo vệ chất lượng kiểm tra, thì phải cẩn thận. Tình trạng này sẽ làm tách rời hai bộ

phận, cảm giác vượt trội của nhân viên phát triển sẽ làm cho nhân viên quản lý chất lượng cảm thấy mình thuộc nhóm thiểu số. Chức năng chính của nhân viên quản lý chất lượng là không ngừng giám định và đánh giá hiện trạng sản phẩm, chất lượng và chức năng sản phẩm có tuân theo mục tiêu đã đề ra không, để cho nhân viên khác tập trung vào nhiệm vụ của họ.

Công việc đánh giá của nhân viên quản lý chất lượng là một mắt xích có tính chỉnh thể, có tính liên tục trong hoạt động phát triển phần mềm. Báo cáo đánh giá tốt về bản chất cần phải phân tích khách quan và có tiêu chuẩn đo lường, như vậy mới có được sản phẩm phần mềm phù hợp nhu cầu thực tế. Không được bỏ qua điểm quan trọng này vì trong quá trình phát triển, nhân viên phát triển có thể vô ý xa rời nhu cầu thực tế do có một số vấn đề phát sinh hoặc do linh cảm nào đó, tạm thời quên mất chức năng cần có của sản phẩm. Trách nhiệm của nhân viên quản lý chất lượng lúc này chính là đảm bảo chất lượng, dùng cách nhìn nhận thực tế, khách quan theo hướng thị trường, không ngừng kiểm nghiệm sản phẩm.

AI THIẾT KẾ SẢN PHẨM?

Nếu giám đốc dự án, giám đốc sản phẩm và nhân viên phát triển không ngừng tranh luận ai có quyền

thiết kế sản phẩm thì đây là nhóm phát triển tồi, chỉ biết để ý đến quyền lực của mình; trong khi đó, uy quyền thực tế bắt nguồn từ việc có nắm vững tình trạng hiện tại hay không. Mục đích thiết kế sản phẩm là đưa những ý tưởng hay nhất vào sản phẩm, cần được kiểm tra thực tế trước khi thực sự bắt đầu dự án. Điều tra thị trường là cách làm tốt nhất, không cần nhiều thời gian và chi phí có thể nhanh chóng dẹp yên tranh luận của mọi người về thiết kế sản phẩm. Dưới đây chúng ta sẽ thảo luận cách tăng sự hiểu biết lẫn nhau, đây cũng là một trong những phương pháp để giải quyết vấn đề này.

Cuộc chiến tranh giành quyền lực sẽ làm lòng dạ con người trở nên hẹp hòi, luôn cạnh tranh chứ không hợp tác.

Tranh luận về thiết kế sản phẩm chính là cuộc chiến giành quyền lực, sẽ làm cho lòng dạ người ta trở nên hẹp hòi, từ đó sẽ không hợp tác được với nhau. Có thể sẽ có người cho rằng sản phẩm còn thiếu chức năng nào đó, kết quả trọng tâm trở thành chức năng chứ không phải bản thân sản phẩm đó. Người lãnh đạo tài năng phải coi tranh luận là triệu chứng có vấn đề của tổ chức, từ đó đi tìm nguồn gốc của vấn đề để làm trọng

tài cho cuộc tranh luận này. Trong phương diện này, kinh nghiệm của tôi lại ngược hẳn với cái được gọi là trí tuệ trong truyền thống, tôi thấy rằng dùng quyền quyết định ai có quyền thiết kế sản phẩm sẽ không có tác dụng gì cả, mặc dù có thay đổi được quyền thiết kế sản phẩm chính thức hoặc có một chút tác dụng gì đó.

Nói thực, bàn về chủ đề ai có quyền, ai phụ trách thật không có gì hứng thú lắm, hơn nữa giải quyết vấn đề này rất mất thời gian. Trong nhóm tổ chức kiện toàn mọi người đều có nhiệm vụ thích hợp; hơn nữa, ngoài việc đảm nhận công việc của mình ra, mỗi người còn đảm nhiệm các vai trò thông thường, điều này cũng được người quản lý đồng tình, do đó mối quan hệ giữa vai trò và năng lực sẽ trở nên hài hoà. Vai trò thông thường sẽ làm cho nhóm hợp nhất, hỗ trợ lẫn nhau. Nhóm được cân bằng, ranh giới quyền giữa mọi người cũng như quan hệ của hàng xóm láng giềng do đó mỗi cá nhân đều được tôn trọng, hiệu quả tổng thể cũng được nâng cao. Tổ chức cân bằng này được hình thành một cách tự nhiên, nếu can thiệp mạnh vào sẽ có những tác dụng phụ. Nếu mối quan hệ giữa các thành viên nhóm không thể hài hoà, người lãnh đạo phải phân tích ra nguyên nhân. Thông thường nếu phân chia lợi ích không công bằng, hoặc nhiều người cùng tranh nhau

một lợi ích, hoặc thù lao không xứng đáng sẽ dẫn đến tình trạng mất cân bằng; đôi khi việc quản lý không linh động làm cho thay đổi nhân sự, hiệu quả cũng sẽ không cao.

Trước khi bắt đầu bất kỳ một dự án nào, giám đốc dự án cần phải hiểu được nên làm rõ vấn đề gì trong nhóm, nhất là nhóm có tính biến động (team dynamics). Mỗi thành viên trong nhóm cần hiểu rõ mình phải làm gì, có những tư liệu gì, mục đích ở đâu giám đốc dự án cũng phải xây dựng nhóm do mình dẫn dắt cần như thế nào, cần làm gì để "kích hoạt" nhóm. Để làm được như vậy, chúng ta bắt đầu vào nguyên tắc đầu tiên.

NGUYÊN TẮC 1: XÂY DỰNG MỤC TIÊU CHUNG

Establish a shared vision

Có thể với bạn điều này không cần phải nói nhiều, nhưng đây chính là vấn đề khá khó khăn. Mỗi thành viên trong nhóm, đều phải hiểu rõ chúng ta cần làm gì, sản phẩm làm ra sẽ như thế nào, chiến lược sản phẩm cơ bản là gì, lúc nào phải hoàn thành. Nếu có cách nhìn nhận khác nhau về mục tiêu chung thì phải thống nhất, chứ không được coi như không có chuyênj gì xảy ra. Nhận thức chung thống nhất là điều tối cần thiết, nếu không phần mềm sẽ không hoạt động tốt, nhiều việc trở nên phức tạp và không giải quyết được.

Có rất nhiều phương pháp xây dựng mục tiêu chung, từ thái cực độc đoán đến thái cực buông lỏng tự do, nhưng trước khi bàn đến các phương pháp này; cần phải nói rõ mục tiêu là gì, bởi hiện nay xây dựng mục tiêu là một trong những điểm để đánh giá năng lực của lãnh đạo. Nói một cách đơn giản, mục tiêu là hi vọng chung, là hình ảnh các sự việc trong tương lai được vẽ ra.

Từ "mục tiêu" hay được các nhà chính trị và nhà kinh tế sử dụng. Bất cứ một nhà lãnh đạo tốt nào đều có trách nhiệm tạo mục tiêu cho những người cấp dưới.

Theo quan điểm của tôi, việc nắm vững và hiểu biết về trạng thái tâm lý của thành viên chính là khởi đầu của mục tiêu, sau đó trong những trạng thái tâm lý phức tạp đó nhà lãnh đạo phải thấu hiểu, uốn nắn và xây dựng điểm chung, đưa đặc điểm cá nhân của người lãnh đạo vào, làm dần mất đi ranh giới phòng vệ giữa lãnh đạo với cấp dưới cũng như giữa các thành viên trong nhóm. Có mục tiêu chung mới tạo dựng được cảm giác quy tụ, cảm giác thống nhất, mọi người sẽ có cảm giác tất cả là một chỉnh thể, đây là nền tảng tâm lý trong không khí làm việc của nhóm.

Cảm giác quy tụ thống nhất, sẽ loại bỏ hành vi chỉ biết có bản thân.

Thông thường, người lãnh đạo có thể lay động tình cảm dựa vào kinh nghiệm chung, dùng các hình thức khác nhau để cho các thành viên biết: quá khứ của chúng ta là thế này, hiện tại là thế này, và tương lai sẽ như thế nào.

Dù có lịch sử chung hay không, "nhà lãnh đạo thành công" có thể tạo ra tiếng nói chung trong nhóm, còn "người kích động quần chúng" lại không làm được như vậy. Khi ý kiến thành viên khác với ý kiến của người lãnh đạo, phản ứng thường gặp là không phản đối trực tiếp, mà: "Đúng rồi, chúng tôi cũng cho là như

vậy, nhưng chúng ta phải làm như thế nào?". Lúc này cần phải nỗ lực, khuyến khích hay thoả hiệp. Người lãnh đạo cần phải tự hỏi: "Nếu như họ lãnh đạo; họ sẽ làm như thế nào? Làm thế nào để chuyển các tâm tư phức tạp của họ thành động lực thành công ?". Những đáp án sẽ giúp người lãnh đạo giải quyết được hầu hết mọi việc.

Nhà lãnh đạo nhìn xa trông rộng có thể nghĩ ra một triển vọng tốt đẹp, mong mọi người chịu khó xây dựng; còn người kích động lại bất mãn với hiện thực. Nhà lãnh đạo nhìn xa trông rộng có thể lãnh đạo thành viên với các suy nghĩ khác nhau nỗ lực hướng về một mục tiêu chung, có khi vứt bỏ những lợi ích trước mắt vì những kế hoạch lâu dài; còn người kích động lại chỉ mong có được lợi ích trước mắt.

Năng lực lãnh đạo và mục tiêu bắt nguồn từ tiếng nói chung giữa lãnh đạo và các thành viên nhóm; nếu không mục tiêu sẽ là không tưởng, và cũng không thể dẫn dắt mọi người đi đến thành công.

CÂU CHUYỆN VỀ BAN NGÔN NGỮ LẬP TRÌNH CỦA MICROSOFT

Tình hình ban ngôn ngữ lập trình của Microsoft năm 1992 không có mấy khả quan. Tháng 4 năm đó,

Microsoft công bố C7. C7 mất hơn hai năm phát triển, nhiều lần không đạt tiến độ và chất lượng sản phẩm kém. Trong khi đó đối thủ của Microsoft lợi dụng cơ hội này, có được đánh giá tốt vượt trên C và C++ của Microsoft. C7 có rất nhiều điểm bất cập không đủ để cạnh tranh, dù nó được đóng gói đẹp, giá hạ, kho hàm số (class library) xuất chúng MFC 1.0 và trình biên dịch (compiler) tuyệt vời của Microsoft. Những ưu điểm này chỉ giúp cho Microsoft duy trì được phần mềm này, nhưng phiên bản nâng cấp không thể không có thay đổi lớn.

Nhóm phát triển C/C++ của Microsoft trải qua nhiều năm bất lợi, riêng giám đốc dự án đã phải thay 3 lần. Denis Gilbert là giám đốc phát triển tài nguyên mới, còn tôi là giám đốc bộ phận kinh doanh kiêm hướng dẫn người sử dụng; tuy trước đây chúng tôi chưa hề lãnh đạo nhóm đông như vậy (hơn hai trăm người), nhưng cả hai đều tràn đầy ý chí tiến thủ, mong làm được việc ra trò. Teff Harbers quản lý nhóm nhỏ AFX, nhóm này có nhiều lập trình viên tinh luyện của Microsoft, phụ trách cung cấp công cụ kho hàm số "xem là được" (What you see is what you get, WYSIWWYG) cho sản phẩm C/C++. Họ có rất nhiều kinh nghiệm đáng quý, đương nhiên sự chênh lệch về văn hoá giữa hai nhóm lúc bắt đầu hợp tác khó tránh khỏi chút va chạm.

Bộ phận ngôn ngữ lập trình là bộ phận lâu đời nhất của Microsoft, sản phẩm đầu tiên là trình biên dịch BASIC do Bill Gates và Paul Allen viết. Khi Microsoft xâm nhập vào lĩnh vực phần mềm hệ điều hành và ứng dụng, tính quan trọng của ngôn ngữ lập trình đã giảm đi đáng kể. Sau khi Microsoft chuyển hướng trọng tâm của mình, tuy sản phẩm ngôn ngữ lập trình vẫn chiếm ưu thế, nhưng các công cụ hỗ trợ người lập trình lại không bằng các đối thủ cạnh tranh.

Năm 1992, hầu hết các cán bộ cao cấp của Microsoft đều mong muốn chấn hưng bộ phận này, có lẽ do họ đã bỏ qua khá lâu rồi, tôi không có ý trách cứ họ, nhưng tôi đoán đa phần là do Microsoft phát triển quá nhanh. Nhưng dù thế nào, điều này cũng làm cho trách nhiệm của tôi và Denis Gilbert càng lớn, chúng tôi càng quyết tâm làm thật tốt nhiệm vụ này.

Áp lực và kỳ vọng đối với chúng tôi không chỉ đến từ cấp cao của Microsoft, mà còn từ những đánh giá của các tạp chí chuyên ngành và trách móc của người sử dụng, thậm chí các đồng nghiệp trong nội bộ Microsoft cũng coi thường bộ phận này. Mọi người đều bảo chúng tôi thiếu mục tiêu, là một đám ô hợp, nếu không thì cũng chê cười chất lượng C7 của chúng tôi, thời hạn hoàn thành thường xuyên hoãn lại. Ngay cả email của Bill Gates cũng kèm theo một câu - bộ phận ngôn ngữ

lập trình là nhóm người dốt nhất Microsoft. Tất cả những điều này không chỉ làm mất mặt, mà chúng tôi cần phải làm gì đó để thay đổi.

Nghĩ về những ngày tháng phát triển và hàng đống công việc của bộ phận này, chúng tôi (chủ yếu là Denis Gilbert) phát hiện bộ phận luôn có vài lỗi sau:

- Không thể hoàn thành nhiệm vụ, do chúng tôi không có đủ năng lực cần thiết.

- Dù có hoàn thành các công việc đúng hạn, thì vẫn mất thị trường. Chúng tôi không tránh khỏi bị giới truyền thông phê bình, hoặc người sử dụng ca thán.

- Tố chất con người và số lượng chúng tôi chỉ làm được 1 dự án, nhiều nhất là hai.

Bước đột phá đầu tiên

Denis hỏi qua ý kiến của tất cả các giám đốc, yêu cầu họ đánh giá về các dự án, phát hiện ra ai cũng cho rằng Caviar là dự án quan trọng nhất. Caviar chính là viết tắt của kế hoạch Visual C++ for Windows 3.1. Chúng tôi kết luận, chỉ cần làm tốt Caviar, lại đưa ra thị trường đúng lúc là sẽ giành chiến thắng trong chiến dịch này, thậm chí còn chiến thắng mãi về sau. Dự án quan trọng thứ hai là Barrauda, thực chất chính là phiên bản Windows NT của Caviar.

Chúng tôi cần lập tức tăng cường hai dự án này, may mắn là cả hai đều có thể hoàn thành, nhưng nếu không làm được thì cũng phải tập trung toàn bộ sức lực vào Caviar. Tất cả tài nguyên của chúng tôi đều tập trung vào Caviar, hy vọng nó thành công chiếm lĩnh thị trường Windows 3.1.

Thế là chúng tôi liệt kê tất cả các dự án từ to đến nhỏ, yêu cầu mọi người chọn dự án mà mình cho là quan trọng nhất, để mong tìm được tiếng nói chung sơ bộ. Nhưng Denis đã không coi trọng nhận thức chung này, thậm chí ông không mời thành viên phát biểu ý kiến bởi ông đã tự quyết định. Chúng tôi hiểu rõ Caviar nhất định phải thắng, dù hi vọng mong manh nhưng mọi người đều nguyện dốc hết sức mình.

Denis triệu tập toàn thể thành viên, nói với mọi người rằng chúng ta cần đánh bại đối thủ cạnh tranh như thế nào. Chúng ta cần phải vứt bỏ các dự án không quan trọng, đồng thời phải tạo lại cơ cấu.

Khởi đầu diễn ra tốt đẹp, nhưng dẫu sao đây cũng chỉ là khởi đầu; các điều kiện quan trọng khác như nhận thức chung, khả năng quyết đoán, mục tiêu rõ ràng, cảm giác thoả mãn về tâm lý của thành viên,... vẫn chưa đạt được. Denis thấy toàn bộ nhóm, khó có thể đánh bại được đối thủ mà cảm thấy đau khổ và cáu giận. Bản thân ông cũng có lực thúc đẩy, muốn chỉnh

đốn tốt mọi việc. Ông cảm nhận được rằng chỉ có đưa kỳ vọng chiến thắng vào trong nhóm, tất cả thành viên mới có thể nói: "Chúng ta phải chiến thắng".

Caviar có thể trở thành Visual C++ cho Windows 3.1

Denis quyết tâm thôi thúc thành viên, Caviar không thể không thắng. Tuy mọi người phải chịu hy sinh cho dự án, và Denis đã gây dựng được điều kiện "cần" để thành công, nhưng vẫn thiếu điều kiện "đủ".

Các vấn đề liên tục nảy sinh

Một đám người chưa hề được tạo dựng thành nhóm chúng tôi không thể vượt bộ phận làm tổ chức trở nên ăn ý nhịp nhàng, hơn nữa phương thức quản lý uỷ quyền ban đầu còn ở trong giai đoạn non nớt. Chúng tôi không có kế hoạch cho các phiên bản nâng cấp, cũng không có quy hoạch kỹ thuật chi tiết, cũng không có kinh nghiệm thực tế để tham khảo. Hầu như ai cũng đánh giá cao khả năng hoàn thành sản phẩm đúng hạn, kể cả đối với mục tiêu mới mà chúng tôi vừa xây dựng. Nhưng dù sao chiến lược "bỏ xe giữ tướng" và việc cải tổ đã thực sự đem lại sức sống mới cho bộ phận của chúng tôi, nhưng kết quả vẫn chưa biết rõ, chúng tôi vẫn chưa biết hoàn toàn có hiệu quả không.

Vài tháng trôi qua, sự thực cho tôi thấy Caviar thực sự không thể hoàn thành đúng hạn, nhưng sự thực cũng cho thấy không thể không hoàn thành đúng hạn. Chúng tôi đã quyết định công bố Visual C++ 1.0 trong triển lãm phát triển phần mềm (Software Development' 93 West) năm 1993, đồng thời tung sản phẩm khắp đường phố để đối thủ cạnh tranh không kịp phản ứng. Đây là mấu chốt thành công của chúng tôi, cũng là mấu chốt để chúng tôi tin tưởng, hạn cuối cùng là 22/02/1993.

Tuy vậy, công việc phát triển phần mềm lúc này như một mớ bòng bong. Chúng tôi không ngừng đưa thêm vào các chức năng linh tinh, điều tốt là chúng tôi không ngừng nâng cao khả năng của hệ thống, điều dở là hầu như không thể xây dựng thành sản phẩm hoàn chỉnh, thậm chí không thể mỗi ngày hoàn thành một tệp tin thực thi. Nhân viên phát triển coi quản lý chất lượng như công việc "hắc ám", mỗi ngày quẳng một mô hình mới cho bên quản lý chất lượng, không cần quan tâm họ phải tốn công sức kiểm tra như thế nào.

Đại thể trước khi tung sản phẩm ra bốn tháng, tôi đã được Harbers cảnh báo, anh ta chân thành cho tôi biết nếu tôi thực sự muốn làm tốt sản phẩm này, cần phải thay đổi một số cách làm. Anh ta giúp tôi hiểu rằng, ngoài tôi ra không ai có thể ngăn tôi làm việc cần

phải làm. Tôi hiểu rõ mình cần phải làm gì, chỉ vì một số lý do (đa số là lý do cá nhân) mà do dự không quyết. Lúc đó tôi là giám đốc kinh doanh, tuy đã lãnh đạo 15 dự án tại bộ phận nghiên cứu phát triển (R&D) của Microsoft, nhưng chưa hề tham gia nhóm C++, nếu tôi không có biểu hiện tốt thì tôi chuẩn bị trở thành vị giám đốc thứ ba phải khăn gói ra đi.

Harbers đã cho tôi thấy mình thật sa sút. Tôi thấy rõ mọi việc đang rất mơ hồ, tôi phải có trách nhiệm thay đổi chứ không phải lẩn tránh. Harbers cũng không né tránh hay che giấu, nói thực vấn đề và cuối cùng những lời nói của anh ta đã làm tôi "tỉnh giấc".

Tôi và Denis thảo luận về khó khăn trước mắt. Thực sự chúng tôi không biết phải làm như thế nào cả, nhưng đều biết rằng không thể không cải tổ một cách triệt để. Chiều hôm đó chúng tôi mở cuộc họp nhỏ, điều này thật không bình thường, bắt hàng đống người tập trung chỉ để công bố một việc rất nhỏ. Đúng vậy, chúng tôi muốn thay đổi suy nghĩ của mọi người.

Để chiến thắng chúng tôi đã cố gắng giảm bớt các gánh nặng không cần thiết, nhưng vẫn thất bại trên thị trường, không so được với các sản phẩm cùng loại của đối thủ. Chúng tôi đã nếm nhiều mùi vị thất bại - thường xuyên làm ra sản phẩm chất lượng kém và không đúng kỳ hạn - mà rõ ràng chúng tôi nhận thấy

mình có thể làm tốt hơn. Chỉ cần chúng tôi đồng tâm hợp sức, chúng tôi sẽ làm được sản phẩm tốt đúng kỳ hạn; không có ai có thể ngăn cản ngoài bản thân chúng tôi (đương nhiên cũng có một số nhân tố không rõ). Tôi càng nghĩ đến hoàn cảnh trước mắt, và ý nghĩa của nó đối với chúng tôi, đối với Microsoft, càng cảm thấy ý chí cao hơn.

Chúng tôi có ý chí, tràn đầy sức sống, chúng tôi có được những điều kiện cần của một nhóm thành công, chỉ thiếu mục tiêu chung. Tôi có ý nghĩ: chúng tôi không chỉ phải đuổi kịp lịch trình, mà còn phải làm ra sản phẩm Visual C++ 1.0 vĩ đại, để cho tất cả mọi người phải ngưỡng mộ, nhất là những nhân viên kinh doanh vốn sợ bán sản phẩm của chúng tôi. Chúng tôi phải thay đổi cục diện này.

Mong muốn chiến thắng của tôi rất lớn, nhưng điều mâu thuẫn là tôi không dám nói cho các đồng nghiệp. Không ai ngăn cản tôi nói ra ý nghĩ của mình, cũng không ai yêu cầu tôi nói ra, lúc này cảm giác cá nhân hầu như không còn quan trọng. Tôi sợ mình bị chê cười: "Anh là ai? Mới đến, lại làm kinh doanh mà đòi dạy chúng tôi viết phần mềm?"

Nhưng nếu có buồn rầu, không làm gì đó cho Microsoft thì chịu không nổi. Chiều hôm đó tôi đã nói cho tất cả thành viên biết cảm nhận của tôi, đồng thời

cũng hỏi xem cảm nhận của họ: "Tôi đã chịu thất bại, các bạn cũng vậy chứ? Tôi đã thấm thía khi làm một giám đốc dự án kém nhất của Microsoft, phê bình của mọi người bên ngoài làm tôi thấy xấu hổ, các bạn có như vậy không? Visual C++ là sản phẩm lớn, đáng tự hào, chẳng nhẽ các bạn không hiểu được điều đó? Tôi biết chúng ta có thể hoàn thành đúng hạn, đồng thời đánh bại đối thủ, chắc không chỉ mình tôi nghĩ thế chứ? Chúng ta sẽ xây dựng nên tương lai cho Microsoft, chúng ta cần phải làm lớn, đúng không?".

Kết quả, hầu như ai cũng có cảm nhận như vậy, tranh nhau phát biểu bổ sung. Cuối cùng tôi đã tổng hợp được mục tiêu chung: Chúng ta phải hoàn thành Visual C++ 1.0 đúng thời hạn, không còn gì quan trọng hơn thế nữa. Tất cả tài nguyên sẽ dồn vào mục tiêu này; các kỹ thuật không cần thiết khác sẽ bỏ đi, các dự án khác (kể cả Visual C++ cho Windows NT) đều tạm hoãn lại, tất cả sức người, tiền bạc và máy móc đều chú trọng vào việc này, đồng thời phải hành động ngay!

Sau cuộc họp tôi yêu cầu mỗi nhóm nhỏ tối thiểu đưa ra năm kiến nghị cụ thể, đề nghị mọi người nêu ra biện pháp hoàn thành Visual C++, đúng ngày 22 tháng 2. Sau đó tôi đã lấy được rất nhiều ý kiến quý báu, trong chương 3 có rất nhiều quan niệm đều lấy từ lần đóng góp ý kiến này. Hơn nữa lúc đó mọi người đều có

quyết tâm hoàn thành mục tiêu, nên chủ động cho người quản lý biết là nên làm gì, sau đó họ tiến hành thực sự. Còn tôi và Denis chịu trách nhiệm đảm bảo môi trường làm việc, chỉ cần có lợi cho thực hiện mục tiêu là chúng tôi đều cố gắng hỗ trợ, thậm chí cả một số cách nghĩ có thể thực hiện mà trái với truyền thống, chẳng hạn phong cách quản lý thành viên lãnh đạo (management led style).

Tôi cảm thấy, trong quá trình thực hiện bất cứ dự án nào đều có một vài "thời điểm mấu chốt", làm cho mọi người có chung một mục tiêu, có tiếng nói chung về mục tiêu đó, có nhận biết rõ ràng về cấp độ ưu tiên của công việc. Sở dĩ nhóm Visual C++ lớn mạnh và lãnh đạo xuất chúng là khi mọi người nhụt chí có thể thay đổi được thái độ của họ.

Trước tiên có người thấy nhóm có vấn đề, liền mạnh dạn nói ra. Harbers dựa vào kinh nghiệm nhiều năm của mình, với những kiến thức độc đáo, đã nhìn ra sự hỗn loạn của chúng tôi; tuy cách làm của chúng tôi về cơ bản là đúng, nhưng chưa đủ. Tuy Harbers không có nghĩa vụ phải đỡ chúng tôi nhưng anh vẫn nói thẳng ra

Trong việc này, Harbers đã khắc phục được nhược điểm của con người. Thông thường rất ít người chủ động giúp đỡ người không yêu cầu giúp đỡ, nhất là

trong việc phát triển sản phẩm trí tuệ. Trước hết, Harbers phải tự tin vào cách nhìn nhận của mình, đồng thời về tâm lý cũng có chút mạo hiểm; sau đó, anh tin tưởng hành động của mình có lợi sự việc trước mắt, đồng thời không sợ mất mối quan hệ với tôi. Để giúp đỡ nhóm, anh không sợ làm tổn thương tôi cũng như cả nhóm.

Lời khuyên thành thực của Harbers làm thương tổn lòng tự tôn của tôi, làm tôi cảm thấy nghi ngờ và không ngừng tranh cãi với anh ta, bảo rằng tôi đã làm sai, cần phải triệt để thay đổi cách làm. Tôi hết sức và tìm mọi cách phản kích lại "sự giúp đỡ" của anh ta. Nhưng anh vẫn chân thành và bình tĩnh để loại bỏ những chống đối vô lý của tôi. Cho đến 4 giờ sáng, tôi vẫn nhớ như in lời nói của anh: "Đừng chú ý đến cách diễn đạt của tôi, đừng chú ý là ai nói với anh, hãy vứt sự kiêu ngạo của anh đi, mà chỉ cần anh hiểu được những gì tôi nói."

Dần dần, tôi thấy mình có thể tiếp nhận lời khuyên của anh, tôi bình tĩnh trở lại, suy nghĩ đã có tính xây dựng. Tôi bắt đầu thấy rằng mình đã quá kiêu ngạo, không nên do tiếp nhận lời khuyên của anh ta mà cảm thấy mình kém hơn. Hiểu được như vậy, tôi đã khôi phục được tính sáng tạo và sức sống mãnh liệt của mình.

Bài học này cho thấy, chúng ta cần thành tâm nói sự thực cho những người "trong cuộc", cho họ biết có vấn đề ở chỗ nào, tiềm năng của họ là gì. Nếu bạn có tấm lòng đủ rộng rãi, khi có người phạm lỗi hãy giúp họ kịp thời sửa đổi, thậm chí đưa ra những ý kiến hữu ích. Nếu lời bạn nói bị cho là quá kiêu ngạo hoặc bị cho rằng cái gì mình cũng biết, hoặc gặp phản đối "Anh là ai? Dựa vào cái gì mà phê bình tôi?", thì bạn cũng không cần quan tâm đến việc phòng vệ của họ, nói thẳng cho họ biết sự việc đang rắc rối ở chỗ nào. Nếu họ nói: "Cái gì, anh cho rằng chỉ có anh hiểu về phần mềm?", thì bạn chỉ cần nói rằng, bạn chỉ muốn cho họ biết thêm một số ý kiến chứ không có ác ý gì, tại sao không thử tìm hiểu xem sao?

Hiệu quả thấu hiểu giản đơn có thể nhanh chóng tiêu tan, bạn cần lấy tấm lòng thực sự với nhau (thậm chí đôi lúc dám chấp nhận mất việc) để nói ra sự thực, như vậy mới loại bỏ tâm lý phòng vệ của đối phương, giúp họ thực tiếp nhận ý kiến của anh.

Cách nói thực này cần có kỹ xảo và trí tuệ cao, nhưng cũng chính là cơ sở để có mục tiêu chung. Bằng cách không ngừng tìm ra sự thật, mới có thể mở được cánh cửa cổ vũ ý chí thành viên.

Tuy nhiên, nói thực lòng cũng phải có người nghe mới được. Sự thực thường tàn khốc, lời nói thật thường

khó nghe, nhưng cho dù bạn phát hiện ra hay ai đó cho bạn biết, bạn đều phải tiếp nhận, những người thiếu sự tự tin hoặc không có cảm giác an toàn thường khó tiếp nhận sự thực, còn phản kháng lại sự thật hoặc có thái độ phản đối sẽ làm cho người ta mất đi sức sáng tạo, hoặc gây ra trực giác nhầm lẫn.

Đây là quá trình có hai giai đoạn, bạn không chỉ học cách tiếp nhận sự thực, mà còn học cách truyền đạt sự thật. Cách truyền đạt sự thực tốt nhất là dựa vào tình cảm, bạn phải quan tâm và có ý tốt thì mới làm cho sự thực được tiếp nhận dễ dàng. Cho dù bạn bị từ chối, bạn cũng đạt được mục đích cổ vũ đối phương.

Chúng tôi cuối cùng có thể hoàn thành Visual C++ đúng hạn, và đó thực sự là sản phẩm tốt; bộ phận tiêu thụ của Microsoft cũng hết lòng hỗ trợ chúng tôi. Thế là chúng tôi đã đánh bại được đối thủ, tuy vẫn còn sức ép và sự uy hiếp nhưng rốt cuộc thì chúng tôi đã chuyển bại thành thắng.

Những ngày sau đó, chúng tôi đã dần lên được kế hoạch khống chế các phiên bản, cứ cách một thời gian nhất định lại đưa ra phiên bản mới, kể cả bản dùng chung (xem thêm nguyên tắc 3: xây dựng kế hoạch kỹ thuật trong phát triển nhiều phiên bản), nhưng việc khó khăn và khó quên nhất, thắp sáng tinh thần nhóm nhất, vẫn là nguyên tắc 1.

NGUYÊN TẮC 2: LÀM CHO MỌI NGƯỜI CHỦ ĐỘNG THAM GIA

Get their heads into the game

Nếu như ai cũng chịu khó suy nghĩ xem làm cách nào để nhóm làm việc có hiệu quả, thì nhóm đó sẽ dễ dàng có được hiệu quả cao, và ngược lại. Đây là vấn đề mọi người thường nói, nhưng trong thực tế, để cho ai cũng động não suy nghĩ thì đã là thành công lớn nhất trong quản lý rồi, điều này cần phải có người dùng trí tuệ và nỗ lực để tạo ra môi trường như vậy.

Tính quan trọng của tư duy mọi người suy nghĩ cho toàn cục vẫn chưa được đánh giá cao, trong khi đó những khó khăn lại bị bỏ qua. Ngoài tư duy nhóm, mỗi người đều có suy nghĩ riêng, không thể nói là sai, nhưng đây chính là nguồn gốc phát sinh các vấn đề.

Mỗi người có suy nghĩ, quan niệm và giá trị quan khác nhau, có thể không bộc lộ ra, cũng không phải là việc gì cũng phải nói ra, nhưng trong nhóm hầu hết mọi việc đều cần mất thời gian để mọi người hiểu nhau, nhất là khi có những suy nghĩ nhạy bén (xem thêm phụ lục). ý kiến của mỗi người đều đáng để lắng nghe, đồng thời những hàm ý bên trong cũng cần bỏ thời gian tìm

hiểu, nhất là khi có lời đơm đặt, trích dẫn cắt xén hoặc hiểu nhầm.

Hành vi của mọi người xuất phát từ tâm niệm của họ, họ có thể thay đổi tâm niệm vì hành vi, nhưng cũng có thể thay đổi hành vi vì tâm niệm.

GIÁ TRỊ CỦA SỨC SÁNG TẠO

Trong môi trường đòi hỏi sức sáng tạo, càng có nhiều biện pháp càng tốt. Bạn chỉ cần khuyến khích mọi người có chung nhận thức này sẽ dễ dàng tạo được ảnh hưởng sâu sắc. Nếu khi thảo luận vấn đề gặp bế tắc, không có biện pháp nào tốt cả, cách làm của bạn là, quy định mỗi người nêu ra tối thiểu một hoặc hai biện pháp, sau đó bạn sẽ thấy không khí sôi động hẳn lên.

Có biện pháp tốt sẽ làm người ta trở nên vui vẻ, nhất là khi hai biện pháp tưởng chừng không liên quan, khi kết hợp lại trở thành chủ ý mới hơn và phong phú hơn, trung khu thần kinh trong đại não sẽ được kích thích. Do đó sức sáng tạo thường đi kèm với niềm vui.

Có người lắng nghe hoặc hiểu được ý kiến của mình cũng làm cho người ta vui vẻ. Đặc biệt là ý kiến của bạn được mọi người sử dụng và tiếp nhận, điều này làm cho tâm lý được vui vẻ và thoả mãn. Nghĩ ra ý kiến là hoạt động nội tại của não, nhưng khi người khác

ngợi khen hoặc khẳng định ý kiến của bạn, có lẽ bạn sẽ bất giác vui sướng mỉm cười.

Biện pháp tốt có tính phát tán cao. Biện pháp tốt sẽ lan truyền rất nhanh, ai nghe thấy đều cảm nhận được niềm vui của tư duy có tính sáng tạo.

Biện pháp tốt có hiệu quả tương hỗ. Khi biện pháp tốt loại bỏ một số giả thiết sai lầm, sẽ có càng nhiều biện pháp tốt được nảy sinh.

Tư duy có tính sáng tạo có thể bồi dưỡng khả năng quan sát và độ nhạy bén của não bộ, cũng như năng lực đánh giá giá trị của biện pháp mới.

Tư duy có tính sáng tạo chỉ có lợi, không có hại. Tuy bồi dưỡng sức sáng tạo đặc biệt nào đó không phải là chuyện dễ, nhưng bản thân việc "nghĩ ra biện pháp hay" rất đơn giản, rèn luyện nhiều sẽ quen, một khi có thời cơ, sức sáng tạo sẽ như "phản ứng hạt nhân" vậy, không ngừng nảy nở.

Nhưng tại sao thành viên lười suy nghĩ hoặc không dám nói ra ý kiến của mình?

- Bởi họ cho rằng không ai coi trọng ý kiến của họ.

- Bởi họ cho rằng tốt nhất là bảo họ làm gì họ sẽ làm.

- Bởi họ cho rằng làm như vậy chẳng được gì, chỉ là sếp thêm đau đầu mà thôi.

- Người quản lý chỉ hô khẩu hiệu thế thôi.

Nếu như ý kiến của mọi người được tiếp thu, họ sẽ chịu suy nghĩ và phát biểu ý kiến. Nếu biện pháp mới của họ được tiếp nhận, có nghĩa là đáng được thực thi, sức mạnh của nhóm sẽ làm tăng thêm các biện pháp hay, điều này được gọi là "Nhất trí trao quyền" (Empowered consensus). Có thể có danh từ khác phù hợp hơn với quan niệm trừu tượng này, nhưng tôi chọn danh từ này do nó biểu đạt được hai quá trình hình thành quyết định: Chủ động tư duy và thử biểu đạt.

NGUYÊN TẮC 3: XÂY DỰNG KẾ HOẠCH KỸ THUẬT PHÁT TRIỂN NHIỀU PHIÊN BẢN

Create a multi-release technology plan

Nếu thành viên vui vẻ tham gia "kế hoạch kỹ thuật", họ sẽ tự tin hơn vào tương lai, một trong những kết quả của nó là kế hoạch kỹ thuật. Trong việc nhất trí trao quyền một cách dân chủ, ai cũng có quyền bỏ phiếu để tham gia vào kế hoạch. Điều quan trọng là, ý kiến của ai cũng được coi trọng. Kế hoạch kỹ thuật là cơ sở cho mọi hành động của nhóm, trong điều kiện lý tưởng nó còn bao hàm ý kiến hay nhất của toàn nhóm.

Kế hoạch kỹ thuật là giao kèo giữa thành viên với công việc của nhóm, đồng thời cũng là chế độ của nhóm. Thông thường mỗi năm sửa đổi một hoặc hai lần, để có thể phản ánh phù hợp ý nguyện của từng thành viên. làm như vậy các thành viên sẽ có cảm giác an toàn, họ sẽ sẵn sàng tin tưởng lẫn nhau và kỳ vọng ở tương lai. Kế hoạch kỹ thuật vẽ ra bức tranh tin tưởng vào tương lai của mọi người, do đó nếu một thành viên cảm thấy không thích công việc hiện được phân công thì không có vấn đề gì cả, đợi lần nâng cấp phiên bản tới anh ta có thể chọn công việc mà mình cảm thấy có hứng thú hơn; hoặc một thành viên không thích hướng đi của sản

phẩm hiện tại, có thể tạm thời lui ra, đợi đến đợt sửa đổi kế hoạch kỹ thuật tới lại tham gia.

Kế hoạch kỹ thuật cũng có lợi cho việc nắm vững lịch trình: Nhân viên phát triển thường muốn đưa thêm một số chức năng khác vào, dẫn đến phần mềm không kịp tiến độ. Nếu có kế hoạch kỹ thuật, nhân viên phát triển sẽ tự tin vào tương lai, biết cách chờ đợi cơ hội, không nói vội nhất thời. Hơn nữa, nhân viên phát triển hiểu rõ cần làm việc gì vào thời gian nào, vừa không phải gánh vác quá nặng, chất lượng sản phẩm cũng tốt hơn. Khi ai cũng biết rõ mình cần phải làm gì trong sản phẩm này, việc kiểm soát vấn đề trở nên đơn giản hơn. Trong đó, quan trọng nhất vẫn là phân chia công việc.

VÍ DỤ THỰC TẾ VỀ MỘT KẾ HOẠCH KỸ THUẬT

Một lần tôi được mời làm cố vấn của nhóm phát triển phần mềm (không phải của Microsoft), giúp họ xây dựng kế hoạch hoạt động. Giám đối dự án xuất thân từ một bộ phận kỹ thuật chính của công ty, đã quyết định tập trung toàn bộ lực lượng vào kế hoạch 5 năm (Dù trong vòng hai, ba năm khoa học kỹ thuật đã thay đổi rất nhiều. Thông thường không nên lên kế hoạch dài hơi như vậy) Khi đó, họ đang thảo luận về những lợi ích do kỹ thuật mới đem lại.

Trong lúc thảo luận có một chủ đề rất hay, đó là một chương trình kỹ thuật thất bại trong quá khứ. Chương trình này đồ sộ nhưng yếu, rất nhiều lỗi, chạy rất chậm, trong đó có một phần mã chương trình đã hơn 10 năm. Các thành viên đều không thích, rất nhiều chỗ không thích hợp với thiết kế hiện tại, thậm chí nhiều người tìm cách lảng tránh cho đỡ phiền phức.

Tôi nghĩ rằng các công ty không ít thì nhiều đều có những chương trình như sau: Tính lôgic kém, không linh hoạt. Các chương trình này trở nên không thể bảo vệ được, càng không thể nói đến chuyện đưa thêm chức năng mới, kết quả làm giảm năng lực sáng tạo của nhóm phát triển.

Tôi gọi tình huống này là "Tuần hoàn ác tính của phát triển phần mềm", khi đưa ra một phiên bản nâng cấp, bạn không thể triệt để sửa chữa chương trình cũ. Ai mà có thể đảm nhiệm lại được việc trải qua cả quá trình phát triển phần mềm, chỉ riêng việc viết lại chương trình đã giảm thấp hoặc ngắt bỏ dịch vụ khách hàng. Khi chương trình chưa linh hoạt hơn và có thêm chức năng mới, có lẽ khách hàng phải mất thời gian tương đồng của vài bản nâng cấp.

Vấn đề tuần hoàn ác tính sâu hơn sẽ là: Tại sao lại có phần mềm kém như vậy? Do không có bộ phận quản lý chất lượng? Hay giám đốc dự án không quan

tâm đến việc phần mềm có thành công hay không? Khi bạn chưa biết lỗi phát sinh ở đâu và tìm cách sửa chữa, thì những vấn đề tương tự vẫn tiếp tục nảy sinh. Lúc này, bạn không thể xây nhà trên nền kiến trúc đã đổ nát được nữa, điều đó chỉ tốn công vô ích!

Trong những nhân viên phát triển có một nhóm nhỏ là nhân viên mới, trưởng nhóm mới và một nhóm lập trình viên mới này phụ trách viết mã cho chương trình; Đương nhiên, họ đã đem lại sức sống và hy vọng mới cho phần mềm. Họ dự định viết lại đoạn mã, nhằm đưa các kỹ thuật cao hơn vào, nâng cao giá trị sử dụng sản phẩm.

Sau một hồi nghiên cứu, nhóm có xu hướng mua thêm kỹ thuật mới về. Tuy nhiên mọi người có hai cách suy nghĩ khác nhau: Những người đồng tình cho rằng tuy mua kỹ thuật mới sẽ gặp rủi ro về tính tương thích, nhưng chương trình cũ rất khó thanh lý, thà bỏ đi còn hơn; Những người phản đối cho rằng nhóm có đủ năng lực để viết lại chương trình, chỉ cần cho họ cơ hội.

Giám đốc dự án cũng không biết nên phải làm như thế nào, về tình cảm thì muốn cho nhân viên chịu khó cải tiến chương trình cũ, nhưng lại không yên tâm là họ có chuẩn bị tốt để tiếp nhận thách thức này không, nhất là khi nghĩ về quá khứ ảm đạm của họ. Cô không cho rằng đây là "lỗi" của những người quản lý

trước, dù họ đã lơ là để cho chương trình cũ "phát triển".

Thông thường, nếu người quản lý không quan tâm đến bồi dưỡng kỹ thuật, nhân tài sẽ bỏ đi, những người kém sẽ ở lại, và sau đó tổ chức và kỹ thuật đều xuống dốc.

Trong ví dụ này, có rất nhiều nhân viên mới (kể cả vị giám đốc kia) đều thấy kỹ thuật đã quá cũ rồi, cần phải loại bỏ. Có lẽ điều mọi người cần là lời hứa của người quản lý về quy hoạch kỹ thuật, tăng sức sáng tạo, phúc lợi của người lao động,... Cô giám đốc hy vọng rằng nỗ lực của mình sẽ thay đổi được tình trạng này.

Cuối cùng, cô quyết định quy hoạch kỹ thuật, đồng thời uỷ quyền cho nhóm phát triển mặc dù biết rằng có thể bị mất chức, quả thực không chỉ làm chương trình cũ được hồi sinh, mà cũng chỉ mất một đoạn thời gian nâng cấp, cũng loại bỏ được nhiều lỗi nhỏ. Trong sự thành công này, một số nhân viên chủ trương mua kỹ thuật bên ngoài cũng bỏ đi.

Khi sửa đổi quy hoạch kỹ thuật lần sau, không chỉ nhóm phát triển được tham gia, mà cô còn tìm cách cho bộ phận chủ quản đào tạo tham gia và hiểu thêm về quyết sách này, để các nhân viên kỹ thuật đảm nhiệm việc hoàn thành công việc đúng hạn có cơ hội thực hiện mong muốn của mình. Cô cũng khuyến khích bộ phận

chủ quản khác tham gia các cuộc thảo luận, khi đó công việc sẽ trở nên suôn sẻ hơn, đồng thời cũng không bỏ lỡ một ý kiến hay nào.

Có rất nhiều cách để quy hoạch kỹ thuật, nhưng chỉ có một nguyên tắc quan trọng nhất. Quy hoạch kỹ thuật tốt sẽ có rất nhiều điểm lợi, trong đó "phân chia công việc" là quan trọng nhất. Quy hoạch kỹ thuật sẽ cho bạn biết mình đi đến đâu, đồng thời biết được nên làm gì, dùng phương pháp gì, tiếp cận mục tiêu như thế nào. Hơn nữa, chỉ có quy hoạch kỹ thuật các phiên bản một cách hoàn thiện mới có thể giúp bạn đánh giá lượng công việc chính xác.

SỬA ĐỔI QUY HOẠCH KỸ THUẬT

Một trường hợp quy hoạch kỹ thuật thực tế mà tôi thấy nữa là ví dụ về tăng cường nhận thức chung. Giám đốc dự án và nhóm phát triển cử ra nhân viên giám sát kỹ thuật để lên bản thảo kế hoạch kỹ thuật trong hai tháng (xem thêm nguyên tắc 5), sau đó đưa bản thảo cho tất cả thành viên đọc, thêm ý kiến, thảo luận góp ý, sau đó gửi cho nhân viên phát triển (khoảng 80 người).

Trong khi đó, các cấp lãnh đạo cũng chuyển cho nhau để đọc duyệt, sau khi tham chiếu với các quy hoạch kỹ thuật khác, sẽ đưa kế hoạch về sản phẩm nhiều phiên bản. Thông thường họ sẽ không sửa các quy hoạch kỹ thuật này, vì họ hiểu rằng đó là kết tinh tâm

huyết của các nhân viên, và cũng là nhận thức chung rất khó mới có thể đạt được. Nhưng họ phát hiện ra một vấn đề: Bản quy hoạch này trông như hàng đống những đặc sắc không liên quan của sản phẩm, những mục tiêu rắc rối như vậy sẽ có hứng thú với nhân viên phát triển không? Làm sao có thể giới thiệu sản phẩm như vậy cho thế giới - Sản phẩm có hàng chục đặc trưng không nổi trội?

Sau đó ban giám đốc tổng kết, đặc trưng sản phẩm này chia làm 5 loại: Chiến lược, cạnh tranh, thoả mãn nhu cầu khách hàng, đầu tư và yêu cầu chức năng có tính điển hình. Chức năng có tính điển hình là đưa các đặc trưng sản phẩm vào n phiên bản liên tiếp một cách có kế hoạch, mục đích cuối cùng là đổi phương thức làm việc của người sử dụng. Đương nhiên theo cách phân loại này không thể nói chức năng có tính điển hình không thuộc cạnh tranh hay chiến lược, hoặc không liên quan gì đến đáp ứng nhu cầu khách hàng, cách phân loại này chỉ để có thể mô tả chính xác hơn mà thôi.

Sản phẩm chiến lược liên quan đến môi trường và những hạn chế cơ bản nhất.

Ví dụ sản phẩm hỗ trợ hệ điều hành nào? sử dụng mô hình đối tượng gì? sản phẩm cần bộ vi xử lý như thế nào? hỗ trợ những ngôn ngữ nào? Ngoài ra, đặc trưng

sản phẩm chiến lược cũng bao hàm chiến lược toàn diện của các sản phẩm khác trong công ty. Ví dụ các hãng bán máy in phải viết trình điều khiển (Driver) hỗ trợ. Sản phẩm cạnh tranh về cơ bản được xây dựng nhằm đối phó với đối thủ cạnh tranh, nhất là những chức năng mà đối thủ có mà mình không có. Tuy không phải cho tất cả các chức năng giống đối thủ vào, nhưng nếu có những chức năng đặc biệt tốt hoặc thu hút sự chú ý của giới truyền thông thì nên đầu tư. Đặc trưng này của sản phẩm có rất nhiều.

Đặc trưng đáp ứng nhu cầu khách hàng rất quen thuộc với mọi người, và khách hàng đều mong muốn. Lắng nghe ý kiến khách hàng, sản phẩm sẽ "gãi đúng chỗ ngứa". Số lượng đặc trưng này nhiều nhất, nhưng chi phí phát triển lại khá thấp.

Đặc trưng đầu tư là đầu tư về mặt kỹ thuật, hiệu ích của nó có thể chưa nhìn thấy rõ nét trong một hay vài phiên bản tới. Đặc trưng đầu tư sẽ có tiềm năng lớn trong tương lai, có thể đem lại tính điển hình cho các phiên bản sau này. Đặc trưng này giúp cho nhân viên phát triển có thể sử dụng đồng vốn đầu tư hiệu quả.

Đặc trưng điển hình có thể thay đổi phương thức làm việc của người dùng. Về cơ bản, đây là mong muốn của hầu hết cac nhân viên phát triển, trong mỗi lần nâng cấp thiết kế để đưa chức năng đặc biệt vào, dần

dần dẫn dắt người dùng. Thông thường đây là những đặc trưng mà bộ phận bán hàng quảng cáo, nó thậm chí có thể thay đổi các quy tắc về cạnh tranh. Có thể nói: Nếu đặc trưng sản phẩm mẫu mực thành công - không có ai có thể cạnh tranh được với bạn, trừ trường hợp sản phẩm của họ cũng có đặc trưng này.

Trong ví dụ này, giám đốc dự án sẽ chỉnh lý lại tất cả các đặc trưng trong quy hoạch kỹ thuật, chia thành 5 loại lớn, và quyết định đầu tư bao nhiêu vào mỗi loại, đồng thời phân tích yêu cầu kỹ thuật tương ứng cần phải thực hiện, đương nhiên phải phù hợp với vốn nội bộ cũng như kỹ thuật được sử dụng. Phần này độc giả hãy tự suy nghĩ, coi như là công việc trong nhà mình, dựa trên các nhân tố như điều kiện của mình, trạng thái cạnh tranh,... để chia sẻ tài nguyên như thế nào cho hợp lý?

Sau đó, trong 5 ngày liên tiếp sau khi nhận lại bản quy hoạch, nhóm phát triển đã thảo luận chi tiết quy hoạch kỹ thuật của phiên bản sắp tới, mỗi ngày họp khoảng 2 tiếng, kế hoạch như vậy đã được duyệt. Trong 5 ngày này cũng là cơ hội cuối cùng để mọi người phát biểu ý kiến (đương nhiên, trên thực tế kế hoạch luôn thay đổi, làm như vậy có thể tăng hiệu quả của kế hoạch). Do lập kế hoạch từ dưới đưa lên, nên không ai phải lo lắng về nhiệm vụ được giao; Do ban giám đốc

tổng hợp các ý kiến để tạo nên đặc trưng sản phẩm hoàn chỉnh, có hệ thống nên mọi người sẽ tiếp nhận mục tiêu chung; Cả quá trình đều công khai và dân chủ, mỗi thành viên đều có quyền của mình, tranh cãi sau cùng sẽ ít, dễ có nhận thức chung, nhiệt huyết cũng sẽ mạnh hơn. Đúng vậy, cả nhóm sẽ nhận ra mục tiêu chung.

Đây là quá trình quy hoạch rất lý tưởng mà tôi đã thấy. Khi viết cuốn này thì sản phẩm của họ đã đưa ra thị trường, mà còn là sản phẩm rất xuất sắc nữa!

Mỗi người đều cần có cảm giác "Sứ mệnh" đối với kỹ thuật, đều có mong muốn tìm kiếm sự tiến bộ, vậy mà những biện pháp mới nếu không phân tích đúng đắn hoặc thiếu thời cơ chín muồi sẽ bị loại khỏi kế hoạch. Các thành viên phải thực sự hiểu nội hàm công việc, và cũng phải được hiểu rằng có khi biện pháp rất hay, nhưng cần phải phân tích kỹ và được thực tế chứng minh là có thể sử dụng và có hiệu ích nhất định, khi đã được kiểm tra chấp nhận, tạo thành nhận thức chung thì cuối cùng sẽ được thực hiện. Như vậy có thể giảm bớt các đề án chưa được suy nghĩ kỹ, rất có lợi cho tính ổn định của sản phẩm.

Việc chia đặc trưng sản phẩm thành 5 loại chiến lược, cạnh tranh, đáp ứng nhu cầu khách hàng, đầu tư và mẫu mực điển hình, nhằm tạo ra hướng quản lý cho

giám đốc, tiện cho việc kiểm tra giám sát việc sử dụng các nguồn tài nguyên cũng như chẩn đoán lỗi. Ví dụ, nếu bỏ qua địa vị dẫn đầu của kỹ thuật, người quản lý phải tăng cường đầu tư cho tính mẫu mực, đồng thời giảm tỷ trọng đầu tư cho các loại khác.

NGUYÊN TẮC 4: HÃY ĐỘNG NÃO

Don't flip the bozo bit

Tôi cần phải nói lại, phần mềm là tài sản trí tuệ. Cần phải vận dụng trí tuệ mới có được sản phẩm phần mềm. Nếu có thể kết hợp tốc độ cao với càng nhiều trí tuệ, giá trị của mỗi phần mềm sẽ càng cao. Đây là sự thực rõ ràng dễ thấy. Đã từng có người hỏi tôi: "Vấn đề gì quan trọng nhất trong ngành công nghiệp phần mềm?"

Tôi đã không do dự trả lời: "Để mọi người suy nghĩ?" Tin hay không là tuỳ bạn. hầu hết mọi người đều không muốn suy nghĩ. Mọi người đều cho rằng họ rất vui khi được suy nghĩ, nhưng thực tế không phải như vậy. Để cho đầu óc trống không thì rất dễ, tại Microsoft chúng tôi gọi đó là bozo, nghĩa là "Đầu đất". Không ai muốn để ý đến những việc làm của họ, dù họ có đóng góp gì thì cũng không có chút sức mạnh nào. Đương nhiên là không thể tin tưởng vào họ, mà điều duy nhất mọi người mong muốn là họ không làm hỏng việc.

Tuy vậy ai cũng có khả năng là "Đầu đất". Bạn thử suy nghĩ xem, bạn có biết mình đang làm gì? Có cảm thấy mình không có một chút năng lực nào? Cẩn thận, "Đầu đất" chính là bạn.

Trong ban của tôi, không ai được phép như vậy cả. Tôi yêu cầu mọi người phải toàn tâm toàn lực cống hiến, mọi người phải luôn bàn về sản phẩm - làm thế nào để cạnh tranh trên thị trường, lúc nào đưa ra phiên bản mới... Đồng thời mọi người phải có cách nhìn nhận thống nhất về sản phẩm, không được mỗi người một ý.

Muốn phán đoán một người có đang suy nghĩ hay không, cách đơn giản nhất là xem anh ta có chú tâm nghe ý kiến của người khác không, đồng thời lập tức đưa ra các phản ứng nội tâm. Đối mặt với những ý kiến tốt hơn của mình, chúng ta cần phải bình tĩnh ngăn chặn tâm lý cạnh tranh hoặc phản ứng phòng vệ, người khác phải tốn rất nhiều công sức trí óc mới nghĩ được như vậy, cần phải phán đoán công bằng những thông tin mới mẻ, có thể rất có giá trị đó.

Người biết suy nghĩ khi bị phê bình hoặc thấy các ý kiến khác tốt hơn của mình, sẽ biết bỏ qua tâm lý tự cao tự đại để đón nhận những thông tin chính xác. Họ có thể tránh được hai hiện tượng tâm lý sau:

Thứ nhất là tâm lý đề phòng, làm cho người nghe không thể tiếp nhận thông tin phê bình. Trong công việc xây dựng sản phẩm trí tuệ cần rất nhiều tư tưởng và ý kiến mới (như đứa con của mình vậy, có một tình cảm rất đặc biệt), người khác khi nghe những ý kiến về sản phẩm hay quá trình chế tạo sẽ cảm thấy như bị chê

bai. Người biết suy nghĩ sau khi cân nhắc cẩn thận, sẽ tự ý bỏ ý kiến chủ quan của mình đi, sau đó tiếp nhận nội hàm chân thực của thông tin, tuy vậy rất ít người như thế này.

Dùng ý kiến của mình để áp đặt cho người khác thì chỉ là kẻ ngốc.

Người không biết suy nghĩ không những không mong người khác chỉ giáo, mà khi người khác có ý tốt đưa ra ý kiến lại đề phòng quá mức, dễ gây ra xung đột, do đó không thể phán đoán chính xác về thông tin nhận được, không có ý thức xây dựng đối với công việc. Nếu hiện tượng này không ngừng xảy ra, người tiếp nhận thông tin có thể có hai phản ứng sau: Một là "Thông tin này thực sự rất quan trọng", hai là "Tên ngốc này đang muốn áp đặt ý kiến của mình đây". Kết quả như thế nào? Đương nhiên là phản ứng thứ hai.

Hiện tượng tâm lý thứ hai là bài xích lẫn nhau, khả năng xảy ra còn cao hơn. Nếu góp ý với đối phương nhưng do sợ thua kém hoặc những tâm lý khác mà từ chối, dần dần, người góp ý sẽ cho rằng đối phương là kẻ ngốc "khó đào tạo".

Trong nhóm mà có người khó đào tạo thì là điều rất nguy hiểm, sẽ có thể phá hỏng mối quan hệ trong nhóm, tạo nên vết thương lớn đối với năng lực của nhóm, mà hầu như không thể bù đắp nổi. Một khi người quản lý thấy rằng một người nào đó không thể hoà hợp với mọi người thì những người khác trong nhóm cũng sẽ cảm nhận như vậy.

Cách tốt nhất là tăng khả năng hợp tác giữa các thành viên, giúp họ khiêm tốn tiếp nhận ý kiến của người khác. Nếu bạn thử đưa ra thông tin mà đối phương không thể tiếp nhận, hãy đổi một phương pháp nhẹ nhàng hơn, tối thiểu thử nói với đối phương rằng kiểu khép kín của anh ta làm bạn khó xử. Ngược lại, nếu có người không ngừng đưa cho bạn những ý kiến "kém cỏi" hoặc "đả kích ác ý", hãy bỏ lòng tự trọng mà suy nghĩ, có phải tâm lý đề phòng đã che lấp hết khả năng phán đoán của bạn. Nếu bạn có thể thực hiện được chuẩn mực này trong nhóm, khi vô tình gây lỗi sẽ có người nhắc bạn, như vậy nhóm mới có thể phát triển hài hoà.

ĐƯỜNG CHẾT

Trong hầu hết các dự án phát triển phần mềm, đa phần khi bắt đầu đều là làm công việc dọn dẹp cho dự

án trước. Nếu dự án của bạn được mở màn bằng phương thức này (Đáng tiếc là hầu hết các dự án phần mềm đều mở màn như vậy), quá trình này được so với "Con đường chết" (death march).

Để kịp tiến độ mà sản sinh ra hàng loạt mã vô bổ khó hiểu.

Khi một dự án bị kéo dài quá lâu (có lẽ do một dự án trước nữa, mà hoãn dự án này lại vài tháng mới bắt đầu), có lẽ giám đốc dự án đã quên điểm cần chú ý, hoặc là khách hàng than phiền quá nhiều, hoặc áp lực của dự án khác đã đến hạn mà ảnh hưởng đến chất lượng phần mềm, hoặc sản phẩm khác khá thuận lợi nhưng nhân viên đã mệt mỏi muốn nghỉ ngơi một thời gian, những lý do này đều dẫn đến sản phẩm ra muộn hoặc kém phẩm chất. Để hoàn thành sản phẩm đúng hạn, các lập trình viên biết rõ bên trong còn có rất nhiều lỗi nhỏ, biết rõ phần mềm được viết còn qua loa, thậm chí còn không dám chắc phần mềm có chạy không, để kịp tiến độ mà hy sinh lòng tự trọng, vứt bỏ lý tưởng về sản phẩm của mình, họ không thể tự hào về sản phẩm, cũng không mong sản phẩm có thể tranh giành chỗ đứng trên thương trường. Trong nhóm phát triển sản phẩm đúng tiến độ và ổn định, mỗi thành viên đều là một anh hùng, vô tư hy sinh cái tôi bản

thân để hoàn thành nhiệm vụ, sự thực cũng như vậy, phát triển phần mềm là thành quả của bao người khổ tâm vắt óc suy nghĩ.

Nhưng bây giờ họ cho rằng mình cần được nghỉ ngơi thích hợp, có phần thưởng thích đáng hoặc cơ hội thăng tiến, hay làm những việc mình ưa thích, hoặc ngồi nghịch máy tính của họ.

Điều quan trọng là dù dùng phương pháp gì, cuối cùng họ cũng hoàn thành phần mềm đúng hạn, có lẽ chưa được mỹ mãn cho lắm nhưng vẫn đạt được mục tiêu. Bất kỳ việc phát triển phần mềm nào có hạn thời gian, thì rốt cuộc ngay cả lập trình viên cũng chỉ quan tâm đến lịch trình còn ngoài ra không để ý đến cái gì khác, đây là một áp lực rất lớn. Sau đó lại bắt đầu một dự án khác, không biết lại phải đầu tư bao nhiêu công sức, điều này vô hình đã tạo ra những lực phản, đáng sợ nhất là tạo ra cảm giác trống rỗng, đây chính là nguy cơ nguy hiểm nhất.

TRỐNG RỖNG

Cảm giác "trống rỗng" của các lập trình viên, cũng giống như bệnh của các công nhân xây dựng trên kênh đào Panama, khi đã phát thì không thể chữa được. Trống rỗng là cảm giác bạn không còn chịu được

áp lực nữa, là sự mệt mỏi và chán nản cực độ, chỉ có những người làm về phần mềm mới có, triệu chứng như sau:

- Nhận thấy phần mềm này đang vắt khô sức lực của mọi người.

- Cảm thấy dự án này được quản lý bừa bãi không thể cứu chữa.

- Vừa nghĩ tới cần ra phiên bản tiếp sau là cảm thấy choáng váng.

- Không thể hoà hợp với mọi người.

- Mất hứng thú với máy tính.

Khi lập trình viên "nhiễm bệnh" này, ngay những tạp chí như "Tuần báo máy tính cá nhân" (PC Week), "Thế giới thông tin" (Infoworld),...họ cũng không muốn xem, cũng cảm thấy các bài viết khoa học thật buồn cười, thậm chí không quan tâm đến những chiếc máy tính mới nhất. Về cơ bản, đó là họ đã dồn hết tâm huyết để làm ra phần mềm tốt nhất, bây giờ đã hết sức hoặc chuyển sức lực sang vấn đề khác, chỉ còn sự chán chường mệt mỏi.

Thực ra, ngay cả người quản lý cũng có thể mắc "bệnh" này, điều này cần phải phòng ngừa, vì nếu người quản lý có triệu chứng này rất dễ truyền cho cả nhóm.

Đối với một nhân viên phát triển phần mềm, yêu thích máy tính là nguồn động lực quan trọng của anh ta. Cũng như bút đối với nhà thơ, mực màu đối với hoạ sĩ, trình biên dịch là tâm linh của người phát triển phần mềm, là công cụ để phát huy tài năng. Khi nhiệt tình đã cạn, nếu dùng tâm hồn trống rỗng để phát triển phần mềm thì cũng như giã tràng xe cát thôi.

Tôi cũng như các nhân viên phần mềm vậy, rất sợ nhiễm phải triệu chứng tổng hợp này. Nếu không tìm cách đề phòng, sẽ làm mất đi sức sống của các nhân viên.

NGUYÊN TẮC 5: THĂM DÒ ĐỐI PHƯƠNG

Use scouts

Trước khi bắt đầu một dự án, hãy đi thăm dò đối phương trước đã!

"Trinh thám" là người được quân đội cử đi để thăm dò tình hình quân địch. Họ phải tìm hiểu các con đường phía trước như môi trường xung quanh, ranh giới hai bên, tài nguyên bên địch, điểm an toàn, đường đi tốt nhất,... Đồng thời luôn chú ý đến bất kỳ hành động nào của đối phương. Lâu dần, khi một nhóm người gặp nguy hiểm trên đường đi, họ cũng cử một người đi xem tình hình con đường phía trước. Nếu dự án phần mềm không có nhiều đoạn đường nguy hiểm thì cũng không cần bàn, ngược lại cũng nên cử "trinh thám" cho dự án.

Nếu chưa có người tìm ra con đường thích hợp, quá trình bắt đầu của các phiên bản chẳng khác gì đi loạn trong sa mạc vậy.

"Trinh thám" cần phải nhạy cảm "đánh hơi" được các vết tích có liên quan: Quan hệ giữa nhóm phát triển với môi trường, các phiên bản mới của hệ điều hành

đang được phát triển, các kỹ thuật có liên quan với dự án (Nếu cần phải mời người nghiên cứu về kỹ thuật này)... "Trinh thám" có thể xây dựng hoặc sửa đổi quy hoạch kỹ thuật của nhiều phiên bản (Xem nguyên tắc 3), họ hỏi thăm khách hàng, học các kỹ thuật có sức cạnh tranh (Tối thiểu phải hiểu cơ bản), đồng thời nghiên cứu quyết định dự án cần thực hiện theo phương thức hay lộ trình nào.

"Trinh thám" phải đưa ra kiến nghị về các yêu cầu phần cứng tối thiểu, phân tích nhu cầu người dùng và quy phạm phát triển phần mềm, chuẩn bị sản phẩm mẫu, đưa ra kế hoạch sản phẩm trong tương lai, đồng thời kiến nghị các đặc trưng trong phiên bản tới của sản phẩm.

Nhiệm vụ và cống hiến của "trinh thám" là vô hạn. Nếu không có người vạch ra con đường đi thích hợp, quá trình phát triển sẽ mất phương hướng, giữa các phiên bản cũng không có hướng rõ ràng. Tuy trinh thám tốt chưa bảo đảm thông tin đầy đủ, nhưng ít ra cũng không đến nỗi bạn bị bịt mắt dẫn đi.

TẦM QUAN TRỌNG CỦA TRINH THÁM.

Gần đây tôi có cơ hội tiếp xúc với ban MIS của một công ty lớn. Trong ba năm qua họ đã chuyển thành công toàn bộ thiết bị đầu cuối của máy chủ sang môi trường mạng và hệ điều hành Windows, PC. Đây là

nhiệm vụ rất khó khăn, cần nhiều nhân viên và tiền bạc, và một thời gian nhất định, cả công ty đã chịu nhiều vất vả trong quá trình này.

Sau khi hoàn thành việc chuyển sang Windows và PC, họ bắt đầu triển khai một bộ phần mềm ứng dụng dạng phân bố để có thể phát huy công dụng của khoa học kỹ thuật mới, đây là một dự án lớn của họ, dự kiến tổng giá thành hạ và lợi nhuận tăng có thể lên tới mỗi năm mấy triệu USD.

Nhóm của họ gồm khoảng 100 người, chủ yếu là lập trình viên và nhân viên bảo đảm chất lượng, hầu hết đều mới vào công ty, hoặc vẫn chưa thành thạo kỹ thuật mới, hơn nữa đây là lần đầu tiên phát triển phần mềm quan trọng, do vậy họ phải khắc phục rất nhiều khó khăn. Khi mới tiếp xúc với họ, tiến độ của dự án bị kéo dài quá mức, đồng thời vượt cả dự toán, hơn nữa cũng sắp hết hạn rồi.

Dự án phần mềm PC này từ khi bắt đầu đến nay đã định 3 năm, môi trường phần cứng, giao thức mạng,... trong quy hoạch đã lỗi thời ngay khi mới phát triển, các công cụ phát triển và hệ điều hành hiện nay đã được nâng cấp 2 lần. Ba năm nay, giá phần cứng giảm mạnh, chức năng PC không ngừng được nâng cao, khả năng xử lý của phần cứng cũng phải nâng lên nhiều lần, máy tính mà họ sử dụng đã hết thời trên thị

trường, nhưng những phần mềm ứng dụng đang dùng thì không thể đào thải.

Thực tế tôi không thể giúp họ giải quyết vấn đề này, chỉ có thể nói tính quan trọng của "trinh thám". Năm đó, họ chỉ dựa vào trực giác và phong cách thận trọng bảo thủ để đưa ra chiến lược về công cụ và phần cứng: "Chúng tôi cố gắng không thay đổi bất cứ thứ gì, chỉ cần dùng được là tốt rồi". Nếu như họ điều tra kỹ về hướng phát triển của máy tính, nếu họ suy nghĩ kỹ về tính tương thích thì họ sẽ biết rằng phải sử dụng công cụ mới nhất và hệ điều hành mới nhất, đồng thời cố gắng dự đoán về phát triển trong tương lai, đầu tư thêm ít tiền sẽ thật xứng đáng, cũng sẽ không đến nỗi vì đưa ra quyết sách sai mà dẫn đến cục diện khó có thể cứu vãn như hiện nay.

Nhưng may mà cuối cùng nhóm MIS cũng thành công, hoàn thành phần mềm mới. Lần này họ đã học được, trước tiên cần cử hai "trinh thám" là những thành viên xuất sắc, điều tra kỹ thuật triệt để và đưa ra quy hoạch hoàn thiện, như vậy sẽ tránh được các nguy cơ trước mắt.

Nói thực, tôi rất đồng tình với công ty như trên, kỹ thuật máy tính phát triển rất nhanh, thách thức những công ty chưa đủ lớn mạnh, hệ thống cũ rất khó vứt đi, kỹ thuật mới lại ép bạn phải tiến lên. Đúng vậy,

sử dụng phiên bản O có lẽ làm bạn chán nản, nhưng từ phiên bản 7 chuyển sang phiên bản 8 có thể có nghĩa là cả một bước tiến lớn. Nếu trước khi ra quyết sách hãy làm tốt công việc trinh thám và quy hoạch hoàn thiện, hơn nữa nếu hiểu được tính quan trọng của nó đối với hệ điều hành, bạn sẽ vui vẻ tiếp nhận những thay đổi do khoa học kỹ thuật mang lại.

Quan niệm cơ bản mà cuốn sách này muốn nhấn mạnh là, chúng ta không có ý làm giảm tốc độ phát triển của khoa học kỹ thuật, cũng không xây dựng hệ thống cứng nhắc, mà giành được lợi ích trong quá trình thay đổi, không ngừng quản lý những biến chuyển mới một cách thích hợp. Hành động vẫn tốt hơn dậm chân tại chỗ, nhóm có sức hành động cao sẽ có năng lực thích nghi với môi trường rất lớn. "Trinh thám" là để chuẩn bị cho những thay đổi của công nghệ, nếu bạn quyết định luôn án binh bất động, bạn sẽ không cần trinh thám.

Đương nhiên, trinh thám cũng có thể gây rắc rối. Nếu họ không có nhận thức rõ ràng về phần mềm nhiều phiên bản, họ sẽ không biết phải đi tìm hiểu cái gì. có thể họ không hiểu được tầm quan trọng của nhiệm vụ này mà thực hiện rất cẩu thả. Trinh thám phải nắm rất rõ nhiệm vụ mà họ đang mang trên vai! Nếu chọn

nhầm hệ điều hành hay công cụ phát triển, có thể sẽ phá hỏng cả một nhóm.

Mặt khác, bản thân nhân viên trinh thám do tầm quan trọng của nhiệm vụ này mà bị thổi phồng bản thân, coi thường những người khác, tự coi mình là người "định nghĩa tương lai". Các thành viên khác trong nhóm vì thế mà bị tl ương tổn, họ không thể nắm được định hướng phát triển, thấy rằng mình không được coi trọng, thậm chí cảm thấy nhiều việc quá làm không xuể, người quản lý lại điều người đi nghiên cứu những kỹ thuật không đâu: "Mỗi tuần tôi phải làm 70 tiếng, còn họ lại ngồi rỗi đọc sách báo". Sự bất mãn này hay xảy ra khi cử người giỏi nhất đi làm trinh thám.

Các thành viên không khỏi ghen tị với công việc của trinh thám. Khi trinh thám tốt hơn những nhân viên phát triển - có vẻ tiên tiến, có vẻ xuất chúng. Làm trinh thám trong vài tháng thật sướng, có thể quan sát các công ty khác, buôn chuyện với các hãng sản xuất, nghịch mô hình phần mềm, có cơ hội phát huy khả năng bản thân, có quyền ảnh hưởng đến tương lai. Nhưng trinh thám không tham gia phát triển phần mềm, chỉ phụ trách quy hoạch sơ bộ và bồi dưỡng nhóm có nhận thức chung. Nếu trinh thám làm cho mọi người tín nhiệm, đồng thời tìm ra được nhận thức chung, nhân viên phát triển sẽ có thể tự làm việc, tin tưởng

rằng đồng nghiệp giỏi làm nhân viên trinh thám có thể tìm ra được đường đi đúng cho mọi người.

Đối với nhóm phát triển, sử dụng trinh thám càng thành công, họ càng tin vào sự lãnh đạo của giám đốc dự án, nhận thức chung của cả nhóm sẽ càng cao; Dù có trong quá trình chuyển đổi kỹ thuật hay không thì đây đều là hiện tượng tốt cho cả nhóm cũng như mỗi cá nhân.

NGUYÊN TẮC 6: CHÚ Ý TỶ LỆ NHÂN VIÊN

Watch the ratio

Một trong những lỗi mà giám đốc dự án hay mắc phải là cho rằng chỉ cần thuê lập trình viên thôi là đủ, các nhân viên khác không cần thiết, hoặc để cho tỷ lệ nhân viên lập trình chiếm tỷ lệ lớn. Có lẽ họ cho rằng lập trình viên càng nhiều, sẽ viết được càng nhiều chương trình, đây là quan niệm rất sai lầm, mục đích của dự án là hoàn thành phần mềm chứ không phải hoàn thành hàng đống mã. Trong nhóm phát triển, thực tế có một số công việc không thích hợp với lập trình viên.

Trong nhóm của chúng tôi, tỷ lệ thường là 6 nhân viên phát triển phần mềm, 2-3 nhân viên bảo đảm chất lượng, 1 giám đốc dự án và 2 nhân viên viết tài liệu kỹ thuật. Trong các ban của Microsoft, tỷ lệ này hơi có thay đổi, cũng có thể không giống với tỷ lệ tại công ty bạn; Nhưng nguyên tắc cơ bản là tỷ lệ giữa lập trình viên và nhân viên đảm bảo chất lượng không quá 2:1.

Thực tế người phụ trách phần mềm hoàn thành đúng hạn lại là nhân viên đảm bảo chất lượng. Khi tiến độ quá chậm, người đầu tiên xem đến là nhân viên đảm bảo chất lượng: Số lượng người có đủ không? Về tiến độ

có phối hợp tốt với nhân viên phát triển? Có thể cảnh báo kịp thời khi có vấn đề? Quan điểm của họ và nhân viên phát triển có khớp? Có phải quá thân với nhân viên phát triển mà nới tay?

Đối với tỷ lệ của nhóm nhân viên cần chú ý đến tỷ lệ số người hữu ích chứ không nên chú ý đến con số thực tế. Nhóm phát triển phần mềm lành mạnh cần có tỷ lệ phù hợp với nguyên tắc trên, trung bình mỗi nhân viên bảo đảm chất lượng không hỗ trợ quá hai nhân viên phát triển: Nhân viên đảm bảo chất lượng suy nghĩ, giám sát và đốc thúc phần mềm có đạt được mức như dự kiến không, còn nhân viên phát triển chuyên tâm viết chương trình và tìm lỗi. Chú ý đến tỷ lệ nhân viên, có thể giúp (hoặc đảm bảo) cho nhóm phát triển cân bằng. Hãy nhớ rằng, cân bằng mới là mục đích của bạn.

NGUYÊN TẮC 7: SỬ DỤNG NHÓM GIÁM SÁT TÍNH NĂNG

Use feature teams

Trong nhóm phát triển C++ của Microsoff, chúng tôi đã sử dụng nhóm giám sát tính năng. Tôi thấy đây là cách làm rất hay, có ảnh hưởng rất tốt đối với chất lượng công việc của cả nhóm.

Nếu bạn hỏi một nhân viên đảm bảo chất lượng công việc của anh ta là gì, anh ta sẽ trả lời: "Giám sát đốc thúc tiến độ phát triển phần mềm, đảm bảo hoàn thành đúng hạn, đồng thời đảm bảo phần mềm đạt được mục tiêu dự kiến".

Anh ta sẽ không trả lời: "*Kiểm thử* chương trình", đây là câu trả lời rất thiển cận. Công việc của nhân viên đảm bảo chất lượng là hoàn thành sản phẩm đúng hạn, là phát triển "sản phẩm", tìm hiểu khách hàng, đồng thời nắm rõ từng công việc trong quy hoạch kỹ thuật đang được tiến hành ra sao. Có thể nói, nhân viên đảm bảo chất lượng phải lo lắng về sản phẩm, thị trường và cả sự nghiệp của chúng ta.

Nhóm của chúng tôi được tổ chức dan xen, trong đó mô hình tổ chức truyền thống là *dọc*, nhóm giám sát

tính năng là *ngang*. Quản trị dự án, đảm bảo chất lượng, phát triển phần mềm, tài liệu kỹ thuật là bốn vai trò khác nhau, thuộc tổ chức phân tầng lớp truyền thống; Ngoài bốn vị trí này, còn cần cử đại diện tham gia nhóm giám sát tính năng, mỗi tính năng đó được làm ra đúng thời hạn, nhóm này có thể họp bất kỳ lúc nào cần thiết.

Phương thức hoạt động của nhóm giám sát tính năng có các nhân tố quan trọng sau: *Trao đủ quyền, trách nhiệm, đồng nhất nhiệm vụ, xây dựng nhận thức chung, địa vị bình đẳng*. Nhóm giám sát tính năng là phương thức tổ chức hữu hiệu nhất mà tôi từng gặp, tôi sẽ thảo luận 5 nhân tố đó dưới đây:

Trao đủ quyền: Biên chế nhóm giám sát tính năng như vậy có vẻ không chính thức, không dễ quản lý như cách phân tầng truyền thống, nhưng nó chỉ phát huy tác dụng khi được trao đủ quyền. Ví dụ nhân viên phát triển, có vẻ như chuyên trách một lĩnh vực kỹ thuật nhất định, nếu cho họ hoàn toàn dẫn dắt việc thiết kế một sản phẩm thì thật không thông minh, nếu thêm một số người ở vị trí khác vào thì sẽ cân bằng hơn nhiều. Chỉ dựa vào quyết định của quản trị dự án, e rằng không đầy đủ, tôi đã xem qua rất nhiều quyết sách tốt (hoặc xấu), nhưng sau đó lại bị quản trị dự án bỏ đi, tôi không hề thấy nhóm giám sát tính năng

nhúng tay vào. Cũng chính là, chỉ cần nhóm giám sát tính năng được trao đủ quyền, và phát huy tác dụng của họ, sẽ đảm bảo chất lượng của quyết sách. (Đương nhiên cần chọn người thích hợp tham gia nhóm giám sát tính năng, xem thêm phần phụ lục).

Trách nhiệm: Tôi cho rằng đây là vấn đề thú vị nhất trong phát triển phần mềm. Chủ đề mà chúng tôi từng thảo luận là "Trách nhiệm của bạn là gì?" Tôi cho rằng, đa số suy nghĩ của mọi người đều không được sử dụng tốt vì mọi người không cho rằng đưa ra biện pháp mới là một trong những trách nhiệm của mình (Xem thêm nguyên tắc 4).

Trước hết, mọi người có thể cho rằng: "Đó không phải là phạm vi trách nhiệm của tôi", do đó không "ôm rơm", không đưa ra bất cứ ý kiến nào có tính xây dựng. Cứ như vậy sẽ có rất nhiều hậu quả, nhất là khi có vấn đề mà không thể dùng lời nói để nối kết mọi người, rất dễ gây ra hiểu lầm, đố kỵ, mà cuối cùng chỉ còn có thể dùng hành động để phản đối.

Chỉ cần mời đối phương đưa ra ý kiến, sẽ dễ dàng loại bỏ tâm lý phòng ngừa.

Sau đó, nếu kiến nghị thực sự nói lên ý kiến của

họ (Có thể phải chấp nhận nguy hiểm), đối tượng bị kiến nghị có thể lại có tâm lý phòng ngừa (Cũng không phải việc của bạn, quan tâm làm gì cho mệt!), gây ra tranh cãi. Có khi tâm lý phòng ngừa không bộc lộ ra, nhất là khi nhóm cho rằng tâm lý phòng ngừa là nhược điểm. Đáng buồn là, càng là nhóm hay cá nhân phát triển, càng có chế độ quản lý tiên tiến, càng có nhiều lý do để chứng minh rằng cách làm hiện có mới là tốt nhất, không cần phải kiến nghị gì cả.

Hơn nữa, nếu người kiến nghị không khắc phục tâm lý đả kích của mình thì kiến nghị đó sẽ không bao giờ vào được lòng đối phương. Tâm lý phòng ngừa rất phổ biến, tâm lý đả kích cũng vậy. Thông thường người kiến nghị đều có một thái độ mang tính công kích hoặc có giá trị phán đoán, đây là yếu điểm của hầu hết mọi người, còn người nhận phê bình là người kiến nghị lập tức kết luận: "Đúng là đồ ngốc", cánh cửa của sự hoà hợp liền được đóng lại.

Rất nhiều ý kiến có tính xây dựng đều tiêu tan như vậy. Tôi nghĩ đến cách duy nhất để giải quyết vấn đề này là hội thảo. Trong hội thảo, mọi người chủ động đề nghị đồng nghiệp nêu ý kiến đóng góp, trong vòng tuần hoàn *đưa - nhận ý kiến, phản hồi thông tin* này ai cũng đưa ra ý kiến, nhận sự góp ý. Như vậy sẽ dễ dàng loại bỏ tâm lý phòng ngừa, chuyển sang nhìn nhận sự

việc chứ không phải nhìn nhận con người, cách truyền thông tin đơn thuần như vậy làm cho ai cũng có thể nhận góp ý và chấp nhận thay đổi. Điều làm tôi ngạc nhiên là, nhờ hội thảo mà các thành viên nắm được các kỹ thuật hoà hợp, và sau đó lại không cần phải hội thảo thường xuyên nữa.

Tôi đã có rất nhiều kinh nghiệm sử dụng hội thảo để giải quyết các vấn đề về phát triển phần mềm. Cách làm giống như nghiên cứu một vấn đề, do một thành viên tình nguyện chủ trì; Vấn đề đưa ra đương nhiên là tình huống thực sự đang phát sinh, thường liên quan đến mối quan hệ trong nhóm, người chủ trì sẽ mô tả tình huống chung, đôi lúc cố ý không nói rõ tên ai đó, chẳng hạn:

- Tôi không thể tiếp nhận cách nghĩ này.

- Không ai xem email của tôi (Chỉ kiến nghị).

- Ai đó không chú tâm vào dự án.

Sau đó, mọi người bắt đầu thảo luận, hỏi người chủ trì các thông tin chi tiết hơn, làm rõ những chỗ còn mơ hồ, hoặc hỏi người chủ trì về cảm nhận đối với các sự việc đó. Sau khi thảo luận làm rõ thì cũng là lúc giải quyết được vấn đề.

Có hai cách làm đơn giản sẽ tạo ra hai kết quả rất quan trọng (Từ trước đến nay chưa hề làm tôi thất

vọng), một là thảo luận như vậy sẽ làm xuất hiện nhiều biện pháp mới: Bạn sẽ thấy người chủ trì không ngừng nói "Cách nghĩ này rất hay", và ghi ngay lại; Hai là kết luận của hội thảo có thể ứng dụng vào các trường hợp khác tương tự, mọi người sẽ thấy, đây là vấn đề mà tháng trước mình đã thảo luận rồi! Khi đã có nhiều vấn đề được giải quyết, mọi người cũng sẽ thấy vấn đề này không có gì đặc biệt cả, và sẽ có cách giải quyết.

Trong nhóm giám sát tính năng, thành công hay thất bại của một người đều rất rõ ràng.

"Trách nhiệm" có rất nhiều điểm tốt, rất ít bị hạn chế khi ứng dụng. Nếu thành viên nhóm lành mạnh, có trách nhiệm với nhau trong cả quá trình thực hiện nhiệm vụ - *thiết kế, phát triển, loại lỗi, chất lượng, kỳ hạn* - sẽ góp ý và phê bình có tính xây dựng, đồng thời tiếp nhận với thái độ đúng đắn. Do mỗi người đều có trách nhiệm giống nhau. Khi đã xây dựng được cảm giác trách nhiệm cho nhóm, nhận thức về trách nhiệm sẽ truyền đến từng người một.

Đồng nhất nhiệm vụ: Trao đủ quyền và cảm giác trách nhiệm, sẽ làm cho người ta có quyền khống chế và ảnh hưởng, làm cho bản thân và nhiệm vụ đồng nhất. Quyền khống chế càng lớn, tính đồng nhất càng

cao. Tính đồng nhất là điều kiện tiên quyết để phát triển bất cứ một phần mềm tốt nào, trạng thái tâm lý có lành mạnh hay không sẽ được nó biểu hiện trên sản phẩm. Ví dụ, có người trong lòng có cảm giác thất bại, sẽ được biểu hiện trong chức năng phần mềm mà anh ta phụ trách.

Trong nhóm giám sát tính năng, ai cũng coi sản phẩm là trách nhiệm của mình, chứ không chỉ hạn chế về mặt kỹ thuật. Không ai có thể che giấu trách nhiệm, thành công hay thất bại của mỗi người đều rất rõ ràng. Bạn không thể trách quản lý không tốt bởi bạn chính là người quản lý, bạn cũng không thể trách nhân viên hỗ trợ các tính năng khác không phối hợp tốt với tính năng sản phẩm của bạn, vì bạn và họ có trách nhiệm tương hỗ. Do đó, bạn có trách nhiệm tìm giải pháp, cũng như có trách nhiệm khắc phục các trở ngại của bản thân.

Xây dựng nhận thức chung: Nhận thức chung là bầu không khí của nhóm giám sát tính năng. Vì nguyên nhân để mọi người tập trung lại là "Tính năng sản phẩm" chứ không phải công dụng, vì trách nhiệm có tính tương hỗ, mở rộng tấm lòng là việc rất cần thiết. Tôi thấy có một số nhóm giám sát tính năng tự xây dựng lại mối quan hệ của họ, xây dựng mục tiêu chung, quy hoạch lại tài nguyên, sửa đổi lịch trình cho thích hợp mà không gây ra xung đột. Cho dù có xung đột, do

không phải đối phó với con người hoặc có mưu đồ riêng nên rất dễ giải quyết, không cần cấp trên nhúng tay vào.

Địa vị bình đẳng: Do mỗi thành viên của nhóm giám sát tính năng có bối cảnh, chuyên môn và vai trò khác nhau, cũng như quan niệm khác nhau, không có trường hợp ai hơn ai nên địa vị của họ đều bình đẳng. Thông thường mọi người đều có cách nghĩ hoàn toàn mới mẻ về các lĩnh vực không thuộc chuyên ngành của mình, kích thích mọi người phá vỡ những hạn chế trong quan niệm cũ, tạo nên những sáng kiến tốt hơn. Do ý kiến của mọi người được nhìn nhận trên các góc độ khác nhau, làm cho công việc phát triển phần mềm được xét trên nhiều nấc mà trở nên toàn diện hơn. Với các vai trò khác nhau họ sẽ quan tâm đến các vấn đề khác nhau:

Quản trị dự án: tình trạng của nhóm hiện nay ra sao? Quá trình phát triển có thuận lợi không? Lãnh đạo có phù hợp và có tác dụng? Chúng ta đã đi đến giai đoạn nào trong chu kỳ phát triển? Tiến độ có được nắm vững? Những chỗ cần các bộ phận khác hỗ trợ đã được hỗ trợ chưa? Mục tiêu có rõ ràng không? Còn các công việc nào chưa hoàn thành? Chủ đề của công việc tuần này là gì?

Nhân viên đảm bảo chất lượng: mình có lấy được

mã chương trình theo giai đoạn như dự kiến từ nhân viên phát triển không? Chương trình có những lỗi nào? Cách đối phó với những lỗi này như thế nào? Còn những tính năng nào chưa được phát triển? Khả năng thực thi của chương trình? Quản trị dự án có biết mức độ ổn định của sản phẩm? Cách làm cho nhóm thấu hiểu lẫn nhau? Có phải ai cũng nhận thức về tình hình sản phẩm? Mục tiêu ngắn hạn là gì?

Nhân viên phát triển: Tuần này có hoàn thành đúng tiến độ công việc; giao mã chương trình có tính giai đoạn cho nhân viên đảm bảo chất lượng và nhân viên tài liệu? Người dùng có cần đến tính năng này? Chương trình này có dễ sử dụng không? Chương trình có chạy đủ nhanh, dung lượng đủ nhỏ? Tôi đã xoá hết lỗi chưa? Có ai bị lỡ tiến độ do phải chờ tiến độ công việc của tôi không? Tôi có hiểu biết mục tiêu ngắn trong tuần? Công việc của tôi có phù hợp hướng của sách lược? Nội hàm công việc có đóng góp cho quy hoạch kỹ thuật hay chỉ là đống rác sớm muộn gì sẽ bị loại bỏ?

Phụ trách sản phẩm/kinh doanh: Tính năng sản phẩm có thu hút khách hàng (Bao gồm khách hàng tiềm năng)? Làm thế nào để mô tả tính năng sản phẩm một cách sinh động? Nó có gây cảm tình cho khách? Sửa sản phẩm như thế nào mới kích thích ham muốn mua hàng? Khi nghe đến sản phẩm của chúng ta,

khách hàng sẽ có những phản ứng gì? Lúc nào khách hàng có thể sử dụng sản phẩm này? Tính năng sản phẩm có tăng mối quan hệ với khách hàng? Giới truyền thông sẽ đưa tin như thế nào về sản phẩm? Mục đích phát triển tính năng này là gì? Tôi có hiểu rõ tính năng và tầm quan trọng của sản phẩm? Có khả năng hoàn thành đúng hạn không? Tôi có cần báo cho công ty biết tính không xác định (hay xác định) của sản phẩm? Cơ hội đưa chức năng sản phẩm ra tại đâu?

Nhân viên tài liệu kỹ thuật/hướng dẫn: Chức năng này có thể sử dụng theo phương thức đơn giản hơn không? Hướng dẫn của mình đã đủ rõ ràng? Đã thiết kế đến mức người dùng không cần học cũng biết sử dụng chưa? Nếu tôi chưa làm xong tài liệu, có quá muộn không? Tính năng sản phẩm đã được nhân viên đảm bảo chất lượng kiểm thử chưa? Cảm giác của mình với tính năng này? Tôi có thể hướng dẫn rõ ràng hơn không? Các thành viên khác có tán đồng tài liệu tôi viết ra không?

Tuy nhóm giám sát tính năng rất quan trọng, nhưng muốn tổ chức thành công thì không phải chuyện dễ. Trong tổ chức phân tầng truyền thống đưa nhóm giám sát tính năng vào sẽ là một chuyển biến rất khó khăn, khó khăn này có thể xuất hiện từ bên trong hay bên ngoài.

Trong nội bộ nhóm giám sát tính năng, bản thân phải đối mặt với tính bất định khá lớn, mọi người không biết phạm trù quyền tự chủ là gì, chỉ biết đó là một nhóm người, cần phải cùng làm việc, nhưng không biết phải dùng phương thức quản lý như thế nào. Bản chất của nhóm giám sát tính năng là phá vỡ vai trò truyền thống, nhưng các thành viên lại thường tự định vị bằng vai trò vốn có của mình, như vậy tác dụng của nhóm giám sát tính năng sẽ bị hạn chế rõ rệt. Nhân viên phát triển thường dẫn dắt hoạt động của các nhóm nhỏ, làm cho các chức năng khác không được phát huy. Như vậy, chức năng của nhóm giám sát tính năng chỉ có thể là tăng sự hiểu biết, điều này hoàn toàn trái với tỷ lệ đầu tư cho họ (Tổ chức, hội thảo, chọn người, hành chính,...)

Trong suy nghĩ của nhiều người, nhóm chẳng qua chỉ là nhận thức chung giả tạo và hoà nhã không thật lòng.

Hơn nữa, thường khi có vấn đề rất nghiêm trọng, họ mới thành lập nhóm giám sát tính năng. Nếu lúc này người quản lý buông tay để mặc họ tự làm tuỳ ý, các thành viên sẽ cảm thấy buồn chán do bị hắt hủi. Đối với những thành viên không tự giác, khi thấy xây

dựng nhóm giám sát tính năng có mức độ cao sẽ cảm thấy trong lòng bất an. Chức năng của nhóm mới này rất khó có thể phát huy, vấn đề do vậy mà càng lún sâu hơn.

Bản thân nhóm giám sát tính năng cũng chứa đầy mâu thuẫn và xung đột, vì trong mắt mọi người, nhóm chẳng qua chỉ là nhận thức chung giả tạo và hoà nhã không thật lòng.

Điều duy nhất không thể phủ nhận của nhóm giám sát tính năng là sáng tạo, với thử thách là những đêm không ngủ, đối mặt với những sợ hãi. Và "xung đột" lại trở thành tiêu chí của nhóm, tuy đó cũng là động lực để phát triển.

Do đó, thời kỳ đầu của dự án thường không nhìn thấy tầm quan trọng của nhóm giám sát tính năng vì chưa có tính thách thức.

Cuối cùng nhóm giám sát tính năng vẫn phát huy tác dụng lớn nhất, chỉ cần được tổ chức thích hợp và trao đủ quyền. Nhóm có tài nguyên, có sáng tạo, được người quản lý hỗ trợ, có thể phát huy sức mạnh kinh người, vào những giây phút quan trọng, họ sẽ phát hiện thấy hoá ra mình có thể làm được nhiều việc rất quan trọng.

Đương nhiên trong nhóm giám sát tính năng cũng

có vấn đề. Rất nhiều bộ óc ưu tú với đầy sức sáng tạo không dám đưa ra suy nghĩ của mình, có một lực cản nội tâm làm họ khó có thể biểu đạt ra. Đa số mọi người đều sợ bị chỉ trích, tâm lý này bắt nguồn từ những kinh nghiệm hồi nhỏ: Do sự hạn chế của bố mẹ mà hầu hết trẻ rất nhạy cảm với các thông tin như "*Không được làm như vậy*", để bảo vệ bản thân, cần cố gắng biểu hiện như mọi người, tuân thủ hệ giá trị chung của đa số những người bình thường mà không dám biểu hiện sự vượt trội, khả năng của bản thân. ***Nhưng, biểu hiện khác người mới là yếu tố thành công trong phát triển phần mềm.***

Hành vi tự hạn chế bản thân, là hiện tượng thông thường trong cuộc sống, nhưng lại là căn bệnh trong môi trường phát triển phần mềm. Thông thường mọi người đều biết cách sống trong hệ thống giai tầng, tự sửa đổi thái độ bản thân cho phù hợp. Nhưng người quản lý cũng có lúc đưa ra tín hiệu "Thôi đi! không được như vậy" (Có thể do các sách lược mù quáng hay sự cố chấp của bản thân, cũng giống như bố mẹ giáo dục hành vi của trẻ), làm cho cấp dưới lo lắng mất nhiều hơn được. Người quản lý cần khuyến khích mọi người nỗ lực phát huy, có thái độ khẳng định và hỗ trợ thành viên xuất sắc, tối thiểu có thái độ cởi mở, tích cực với những ý kiến bất đồng.

Mặt khác, mỗi người sẽ có phản ứng khác nhau với ý kiến này. Mỗi người cảm nhận được mình được trao đủ quyền, gánh trách nhiệm lớn, sẽ tự truyền tâm tư này cho các thành viên khác. Sự tự do cá nhân này được biểu hiện khác nhau theo từng thời cơ, và cũng được nhấn mạnh theo các phương thức khác nhau, không ai giống ai, nhưng rất khó nhận ra. Ai cũng mong muốn được thách thức, được khuyến khích, được đào tạo; Sức bật và tính nhẫn nại giúp mọi người tạo nên đoàn thể, cùng nhau trưởng thành trên chính đôi chân của mình. Quá trình trưởng thành này tuy rất gian khổ nhưng lại là điều kiện tiên quyết của nhóm làm công việc trí tuệ xuất sắc.

Trong nhóm giám sát tính năng dễ xảy ra đối thoại rất lạ, như kiểu ai có thể quyết định vấn đề gì, ngoài việc kiểm soát ra người quản lý còn đóng vai trò gì.

Tôi cũng chú ý thấy, mức độ tham gia vào nhóm giám sát tính năng, tỷ lệ nghịch với mức độ thành công trong tổ chức chức năng cũ (Ví dụ vai trò *tài liệu hệ thống*) của họ. Khi chức năng của cả tổ chức biểu hiện không nổi trội, trong đó có một bộ phận chức năng tạm ổn, làm cho cả ban có biểu hiện nổi trội, sau khi cải thiện chức năng của cả tổ chức, cảm giác rời rạc ban

đầu lại làm cho người của bộ phận này khó có thể hoà nhập vào nhóm giám sát tính năng.

Chúng ta bàn tiếp đến các vấn đề ban ngoài mà nhóm giám sát tính năng có thể gặp phải. Người tham gia thực tế trực thuộc bộ phận chức năng (Như quản lý dự án, đảm bảo chất lượng,...) có mục tiêu riêng của bộ phận anh ta, khi điều sang nhóm giám sát tính năng, đương nhiên sẽ có chút phiền phức. Có lẽ bộ phận cũ không sáng tạo như vậy, nhưng trước khi thành lập nhóm giám sát, họ có cống hiến nhất định với tổ chức, không ít thì nhiều có thái độ đối lập với việc thành lập nhóm giám sát chức năng, trong lĩnh vực chuyên môn cũ họ cũng xây dựng cảm giác trách nhiệm và chỗ đứng nhất định, bây giờ đưa họ vào một tổ chức mới, làm những công việc không đúng sở trường, hơn nữa trong môi trường cũ chưa được trao đủ quyền, họ chỉ dựa vào năng lực để giành chỗ đứng, vậy mà lại không cần đến, bây giờ trong môi trường mới được trao quyền nhiều hơn, làm cho họ nghi ngờ về giá trị của những thành công ban đầu, do đó mà họ càng khó thích nghi với môi trường mới.

Nhà quản lý biết phải dùng phương thức truyền thống như thế nào để tung ra sản phẩm, khi nhìn thấy những công việc mới bắt đầu của nhóm giám sát tính năng khi mới thành lập, cảm thấy họ rất ấu trĩ và lo

lắng "Họ làm thế thì sai rồi, nên để tôi làm, hoặc để tôi hướng dẫn cho". Đây đúng là quan tâm thật tốt, người quản lý không quên trách nhiệm của mình với sản phẩm, nghi ngờ nhóm giám sát tính năng bị "*chập*" (Bản thân tôi đã từng nghĩ như vậy), ai lại để những người không có kinh nghiệm quyết định hướng phát triển của kỹ thuật và sản phẩm, đồng thời phạm lại lỗi mà người quản lý trước đây đã gặp!

Người quản lý giữ cân bằng giữa "khuyến khích" và "kiểm soát" cấp dưới, nên xuất hiện đối thoại rất lạ, như hiểu ai có thể quyết định vấn đề gì? Ngoài việc kiểm soát ra còn cần đóng vai trò quản lý gì? Nếu không phải quyết định của người quản lý, anh ta sẽ phụ trách như thế nào? Định vị như thế nào?

Năm đó tôi đã từng hoài nghi, cũng như các vị giám đốc khác nghi ngờ chúng tôi tại sao phải thành lập nhóm giám sát tính năng. Tôi đã từng nửa đêm bật dậy, tự hỏi mình tại sao phải tự rước họa vào thân, nghĩ ra biện pháp điên rồ làm cho ai cũng phải dồn hết công sức. Dần dần, chúng tôi hiểu rằng vai trò của nhà quản lý cần phải là hướng dẫn, thách thức, khích lệ, và khẳng định quá trình sáng tạo này. Nếu quan điểm của chúng tôi đúng, thực tế sẽ tự chứng minh. Khi trao quyền cho nhóm giám sát tính năng, trên tiền đề giám đốc thách thức các thành viên để họ tự kiểm tra xem

xét động cơ và hành vi của mình, hỗ trợ họ đạt được nhận thức chung và hoá giải các mâu thuẫn xung đột, giúp họ thấy rõ người quản lý rất hiểu họ, sẽ ủng hộ quyết định của họ. Chúng tôi luyện cho các thành viên tự đi tìm hiệu năng, đây là yêu cầu rất cơ bản, vì chúng tôi muốn thành viên tự chịu trách nhiệm, cho họ có năng lực quyết định và thực hiện theo phương thức lý tưởng nhất của họ.

Uy quyền đến từ học thức, chứ không phải địa vị.

Đương nhiên, cả quá trình này là tuần tự, xét trên một ý nghĩa nào đó, nó vẫn không ngừng phát triển. Nhiều lúc, nhóm mong muốn giám đốc trực tiếp quyết định và xây dựng mục tiêu cho họ; Cũng có lúc giám đốc yêu cầu họ làm. Tổ chức theo chức năng truyền thống và nhóm giám sát tính năng không có ranh giới rõ rệt. Nhưng tất cả đang thay đổi, tôi cảm nhận thấy mọi người đang đi đúng hướng của mình.

Trong một dự án lý tưởng, về cơ bản tồn tại hai vai trò: Người sáng tạo và người thúc đẩy. Người sáng tạo là những nhân viên chuyên nghiệp, như phát triển phần mềm, tiêu thụ, bảo đảm chất lượng và viết tài liệu; Còn người thúc đẩy phụ trách tập trung nhận thức

chung và duy trì môi trường phát triển tốt nhất. Tại đây có thể phát huy sức sáng tạo, có đủ tài nguyên để giải quyết các vấn đề và thực hiện lý tưởng, vận hành của tổ chức rất có hiệu quả. Người thúc đẩy chính là quản trị dự án hoặc quản lý, năng lực của họ không hiển thị trực tiếp trên sản phẩm, nhưng lại là nguồn động lực thúc đẩy sản phẩm thành công.

Người thúc đẩy cũng như người sáng tạo phải phụ trách kết quả cuối cùng, còn người sáng tạo cũng phải như người thúc đẩy phụ trách nhận thức chung của nhóm. Làm trách nhiệm hai vai trò tương hỗ lẫn nhau là cách làm rất tốt, có thể bổ sung hỗ trợ lẫn nhau. Tính quan trọng của tổ chức tầng lớp truyền thống dần giảm thấp, quyền lực đến từ trí tuệ chứ không phải địa vị, đây là đột phá có tính cách mạng, đáng để các tổ chức phát triển phần mềm suy nghĩ.

NGUYÊN TẮC 8: TRÁCH NHIỆM CỦA QUẢN TRỊ DỰ ÁN

Use program managers

Quản trị dự án là một thành viên của nhóm phát triển, có các trách nhiệm sau:

- Lãnh đạo mọi người làm ra một sản phẩm thành công.

- Dẫn dắt mọi người có kỳ vọng và niềm tin sâu sắc với sản phẩm.

- Lãnh đạo nhóm thực hiện ý tưởng, đưa sản phẩm dự kiến ra đời.

Muốn định nghĩa "Quản trị dự án là gì", không bằng định nghĩa "Quản trị dự án không là gì" sẽ đơn giản hơn. Vai trò thực tế của quản trị dự án không phải là tính chất công việc mà như trực giác của nhiều người thường nghĩ.

Quản trị dự án không hề có (hoặc có rất ít) quyền lực hay địa vị chính thức, tối thiểu là lúc đầu như vậy, do đó quản trị dự án hay suy nghĩ về vấn đề này. Quản trị dự án kém sẽ nghĩ rằng công việc của anh ta là vẽ ra quy cách phần mềm để cho những người khác viết chương trình, kiểm thử, viết tài liệu,... sau đó hoàn

thành sản phẩm theo lý tưởng của mình. Trong trường hợp tốt, mọi người cho rằng anh ta ngây thơ; Trong trường hợp xấu, mọi người cho rằng anh ta ngu dốt, làm không được việc. Trước khi quản trị dự án có giá trị đối với nhóm, không nên có bất cứ quyền giám sát trực tiếp nào; May mà một nhóm phát triển lành mạnh không bao giờ để chuyện này xảy ra, thành viên sẽ ngăn chặn đúng lúc những giám sát trực tiếp bất hợp lý của quản trị dự án.

Đương nhiên những việc này có thể làm quản trị dự án phẫn nộ, do đó yêu cầu cấp trên cho mình nhiều quyền hơn, nếu cấp trên dại đến mức nghe theo anh ta như vậy, sẽ càng dẫn đến những chuyện dại dột hơn (xem nguyên tắc 9).

Thường chỉ khi tình hình tồi tệ đến mức không thể cứu vãn được nữa, quản trị dự án mới hiểu ra "chính trị" chỉ là cái cớ hợp pháp.

Tôi đã gặp được nhiều loại phản ứng khác nhau từ cảm giác thương tổn của quản trị dự án. Thường gặp nhất là quản trị dự án bắt đầu sử dụng vai trò thực sự của mình - lãnh đạo - để có những hành vi tiêu cực, biểu hiện là "Rác chính trị" (Political bullshit) ác ý. Ngoài việc phải có kiến thức nhất định về khoa học kỹ

thuật, lãnh đạo nhóm phát triển phần mềm cần có khả năng quan sát nhạy bén cao độ, cần phải nhận ra các tâm tư ẩn dấu sau các hành vi của nhóm. Có thể thấy, một số quản trị muốn tránh những vấn đề phiền phức của con người, tìm cách đơn giản hơn để nâng cao bản thân, mà trong công ty phần mềm không tránh khỏi khoa học kỹ thuật nắm giữ văn hoá, rất dễ cho rằng "chính trị" là con đường tiến. Đương nhiên, điều khiển kỹ thuật sẽ đơn giản hơn con người. Nhưng khi mọi người than phiền về những mưu mô chính trị, thực ra là có ác cảm với lãnh đạo. Đáng buồn là cảnh bất mãn này đâu đâu cũng có, tổ chức lành mạnh sẽ ít hơn nhiều. Do đó quản trị dự án thất bại chuyển lỗi sang hai chữ "chính trị". Nhưng chỉ đến khi tình hình tồi tệ đến mức không thể cứu vãn nổi nữa, quản trị dự án mới hiểu ra "chính trị" chỉ là cái cớ hợp pháp, lẽ ra phải lãnh đạo thay vì giám sát, mới hiểu ra trách nhiệm của bản thân. Có ai muốn ở trong một tổ chức hỗn độn? Khi quản trị dự án hay ai đó làm tổn thương tính lành mạnh của tổ chức, đó cùng là lúc đón nhận sự thất bại.

Tuy lúc mới đầu quản trị dự án chưa xác định được vai trò của mình, không dám khẳng định cống hiến của mình cho dự án, nhưng khi nhận ra nên dùng lãnh đạo thay cho giám sát khống chế, sẽ cảm thấy tất cả đều hợp lý, muốn sao được vậy. Quản trị dự án cần

lãnh đạo theo phương thức giáo dục để dẫn dắt nhóm phát triển.

Quản trị dự án cần có bối cảnh kỹ thuật, đồng thời phải nắm vững hai vấn đề: Một là nắm vững kỹ thuật dùng để phát triển sản phẩm; Hai là có khả năng lãnh đạo kỹ thuật xây dựng phần mềm, đây cũng chính là phạm vi thảo luận chính của cuốn sách này. Quản trị dự án cần phải biết kích thích, khích lệ, yêu cầu nhóm làm ra những sản phẩm tốt nhất và có hiệu năng công việc tốt nhất; Nắm vững từng hạng mục đầu tư và chi tiết tạo ra trong quá trình chế tạo phần mềm; Biết cách dùng phương thức tốt nhất định nghĩa sản phẩm và duy trì kỹ thuật lành mạnh. Sau cùng, quản trị dự án phải là phát ngôn viên của nhóm, đối mặt với truyền thông, khách hàng và cả tổ chức.

Quản trị dự án là người quan trọng duy trì linh hồn nhóm, phải giỏi thấu hiểu và lắng nghe người khác, có bản lĩnh suy nghĩ trên địa vị của người khác. Tóm lại, quản trị dự án là nhân vật trọng tâm trong phát triển phần mềm.

TINH THẦN ĐỒNG ĐỘI

Ở đây tôi nhấn mạnh một quan niệm, mà sau này tôi cũng đề cập: Phần mềm đại diện cho nhóm sáng tạo

ra nó. Bạn muốn hiểu nhóm này? Xem phần mềm của họ là biết ngay, hoặc ngược lại. Tôi nhấn mạnh quan niệm này vì đây là cơ sở quản lý phát triển phần mềm. Thường khó có thể phán đoán một nhóm dựa vào một thời điểm, thời đoạn, hay dựa vào hành vi của họ, nhưng hoàn toàn có thể dựa vào phần mềm. Phần mềm sẽ biểu hiện trung thực nhóm sáng tạo ra nó: Tất cả ưu điểm hay nhược điểm, các lỗi tiềm ẩn hay sự đoàn kết. Nếu muốn hỏi bất cứ điều gì về nhóm? phần mềm của họ chính là câu trả lời. Nếu đối tượng để hỏi là người, bạn còn cần chú ý nét mặt và các cử chỉ chân tay, nhưng nếu bạn nghiên cứu kỹ phần mềm của họ, bạn không cần phải hỏi ai cũng biết nhóm của họ có vấn đề hay không. Phần mềm sẽ tự biểu đạt, nó không nói dối bao giờ.

Lý luận này là cơ sở quản lý phát triển phần mềm, có thể dùng đẳng thức sau:

Nhóm = Phần mềm
(Team = Software)

Nguyên tắc cơ bản là: Nếu bạn muốn tìm hiểu bất cứ điều gì của nhóm, xem phần mềm của họ là được; Nếu thông tin khớp nhau, bạn có thể tin tưởng vào tình trạng nhóm mà mình nhìn thấy, ngược lại, phần mềm chưa biểu hiện đúng tiêu chuẩn cần có, có nghĩa là nhóm có vấn đề, dù nhóm xem có vẻ lành mạnh, bạn

cùng cần phải nghiên cứu tìm ra vấn đề để giải quyết triệt để.

Nhóm tương đương với phần mềm, đúng vậy, nhưng nhóm là gì? Bạn dùng phương thức và phong cách gì để lãnh đạo nhóm? Tôi tin rằng mấu chốt của phát triển phần mềm có quan hệ chặt chẽ với "Tinh thần đồng đội". Nói như vậy thật trừu tượng, dưới đây là mô tả cụ thể của "Tinh thần đồng đội":

- Một nhóm người đồng tâm hiệp lực, tập trung trí óc mọi người, cùng sáng tạo tài sản trí tuệ.

- Sức sáng tạo cá nhân là điều kỳ diệu, bắt nguồn từ tiềm năng tâm trí con người, được tình cảm phát triển, bị kỹ thuật trói buộc.

- Một nhóm người toàn tâm toàn ý cống hiến sức sáng tạo của bản thân, kết hợp thành một sức mạnh rất lớn. Sức sáng tạo được kết hợp là do họ có quan hệ tương hỗ, do đó càng trở nên phức tạp.

- Trong tình huống phức tạp này, lãnh đạo trở thành chỉ huy "Dàn nhạc" con người, dẫn dắt kết nối mọi người lại với nhau.

- Khi sự kết nối chính xác và lành mạnh, có thể kết hợp hoàn toàn lực lượng cả nhóm, tạo nên những hiệu quả tương ứng, loại bỏ hiềm khích. Kết nối thấu hiểu lẫn nhau làm cho tư tưởng được giao lưu truyền đạt rộng rãi trong nhóm.

Phẩm chất công việc của nhóm còn quan trọng hơn cả lịch trình, còn sự vĩ đại của tác phẩm chỉ được thực hiện khi đặc biệt tăng cường "Tinh thần đồng đội". Có thể coi "Tinh thần đồng đội" như giá trị bình quân tinh thần của mỗi thành viên, mà tinh thần cá nhân làm cho anh ta biết suy nghĩ, biết cảm giác và biết suy đoán.

- Nếu coi thường "Tinh thần đồng đội" sẽ nhận được kết quả tầm thường.

Vậy cần có thái độ thế nào về "Tinh thần đồng đội"? Đáp án chính ở trong cuốn sách này.

NGUYÊN TẮC 9: UY QUYỀN CHỨ KHÔNG PHẢI BẠO QUYỀN

Be an authority, not an authority figure

Trong tổ chức, đa số mọi người đi tìm uy quyền cho mình, dù đó là tốt hay xấu. Hiện tượng tâm lý này có thể do họ mong muốn một kết cấu quyền lực như trong gia đình: Bố mẹ khống chế và bảo vệ trẻ, còn trẻ dựa vào bố mẹ. Dù nhu cầu này bắt nguồn từ đâu mọi người đều mong có quyền lực và đối tượng dựa dẫm, và coi người lãnh đạo là hoá thân của quyền lực - có lẽ do người quản lý cũng có mong muốn này.

Mục đích của uy quyền là để cho mỗi thành viên có uy quyền nghề nghiệp của mình, tự tin với nghề nghiệp của nhóm, đó mới thực sự là uy quyền của người quản lý.

Trong bất kỳ xã hội nào, mọi người thường coi đoàn thể là một kết cấu đã quen thuộc, xu hướng này rất rõ trong nhóm làm việc. Mọi người thường tự coi những người gặp trong công việc, như những người đã gặp trước đây, nhất là việc hay coi người quản lý như cha mẹ, thậm chí, người xuất sắc trong nhóm có thể coi lãnh đạo cấp trên như một nhân vật đầy uy quyền.

Hiện tượng này có thể tạo ra hình tượng "bạo quyền" của người quản lý.

Cũng như ở nhà vậy, một tổ chức lãnh mạnh phải trải qua 3 giai đoạn: *Trẻ, thanh xuân và trưởng thành*. Vào thời kỳ trẻ khi mới thành lập nhóm, thành viên bất giác coi người quản lý như người rất có uy quyền: Quyết định mọi việc, đảm bảo tài nguyên, quyết định thưởng phạt. Còn các thành viên tự xây dựng vai trò cho mình: Phục tùng. Do họ tin tưởng vào uy quyền, tâm lý thành viên sẽ đưa hàng loạt mô hình "lẽ ra phải như vậy" vào hình tượng người quản lý, và coi quan hệ giữa quản lý và nhóm rất đơn giản và ngây thơ. Loại mô hình quản lý "lẽ ra phải như vậy" bao gồm đặc tính, uy quyền, thành công... của lãnh đạo. Thực ra điều này không có ý nghĩa gì trong thực tế, quản lý thông minh sẽ không lãng phí thời gian tìm hiểu uy quyền của mình trong lòng mỗi người.

Mọi người có khuynh hướng coi người quản lý có uy quyền như mẹ vậy.

Nếu quản lý tiếp nhận hình tượng uy quyền một cách ấu trĩ như vậy, với quan hệ không lành mạnh "lãnh đạo - thành viên", cả nhóm sẽ mãi ở giai đoạn trẻ. Hãy nhớ rằng, mục tiêu của chúng ta là làm cho mỗi

thành viên đều có uy quyền, chứ không phải tập trung uy quyền cho quản lý. Người quản lý phải tốn thời gian mới có thể giúp mọi người chấp nhận quan niệm này, có lẽ sẽ có người than phiền rằng trong nhóm phải có quan niệm đúng đắn về uy quyền thì mới có thể phát triển được. Đối với nhóm chưa trưởng thành, phác hoạ mục tiêu còn lễ hơn là tạo nhận thức chung, bạn có thể tập trung mọi người lại, nói cho họ biết mục tiêu bạn xây dựng cho họ là gì, đây là cách làm ngắn hạn, lâu dần sẽ xuất hiện những chuyên gia về kỹ thuật hay sản phẩm, hướng dẫn mục tiêu của mọi người đối với sản phẩm, người quản lý phải tin tưởng và ủng hộ mục tiêu này, đồng thời truyền ra cả nhóm.

TRAO ĐỦ QUYỀN

Tôi rất muốn tìm một hàm nghĩa khác để mô tả "Trao đủ quyền" bởi hiện nay từ này đã bị lạm dụng quá nhiều. Nhưng có dùng danh từ gì biểu đạt, quan niệm này vẫn mãi là giá trị trung tâm của nhóm chế tạo sản phẩm trí tuệ. Mọi người thường không phân biệt rõ giữa "Cho phép" và "Trao quyền", cho phép là "Cho thành viên làm bất cứ việc gì mà anh ta cho là tốt nhất", còn trao quyền là "Để thành viên suy nghĩ, sau đó dốc hết sức làm", hai vấn đề hoàn toàn khác biệt.

Trao đủ quyền làm cho thành viên có thể phát huy khả năng, không bị cản trở, giúp họ loại bỏ tất cả

chướng ngại, để họ dựa vào sức lực bản thân đạt được thành công. Tự do là nền tảng của trao quyền, để họ tự do phán đoán và thực hiện, tự do suy nghĩ và biểu đạt những việc họ cho rằng nên nghĩ và nên biểu đạt, tự do chấp nhận mạo hiểm mà không sợ bị phạt. Trao đủ quyền là kết quả học tập đầy đủ chứ không phải là bỏ qua, để mặc họ muốn làm gì thì làm. Khi quản lý nói với cấp dưới "Đây là quyết định của cậu", thì anh ta đã phải hỗ trợ đầy đủ - huấn luyện, thông tin, tài nguyên - để cấp dưới có thể đưa ra quyết định đúng đắn, đó mới là trao đủ quyền. Nếu không, để cấp dưới quyết định thì chẳng khác gì không thèm quan tâm đến họ.

Nếu ai cũng được trao đủ quyền, vậy khi xung đột thì phải quyết định như thế nào? Đây là vấn đề thuộc về lý luận, trong thực tế sẽ không xảy ra. Trong môi trường trao đủ quyền, không phải là hỗn độn vô tổ chức mà là sự thực và tài năng sẽ lãnh đạo tất cả, ai cũng có cảm giác an toàn, do đó không còn thái độ hẹp hòi hay kiêu ngạo, và thế là việc thiết kế phát triển phần mềm trở thành việc sử dụng tài nguyên một cách đơn thuần. Nhóm được trao đủ quyền sẽ có khả năng phân tích ưu nhược điểm của các phương pháp, đồng thời chọn ra phương pháp có lợi cho mục tiêu, quyết sách không hề đúng hay sai mà là cân bằng giữa chức năng sản phẩm, tài nguyên và thời gian.

Bạn biết cách tổ chức nhóm, dẫn dắt nhóm phát triển, do cung cấp môi trường kỹ thuật tốt, trao quyền cho nhóm hoàn thành sản phẩm đúng hạn, đã phát triển nay còn phát triển hơn, tất cả đều là uy quyền của bạn. Kiến thức của bạn là "tinh thần đồng đội", "tác phẩm đồng dội" hay quan hệ giữa chúng. Uy quyền thực sự của bạn đến từ những việc này, chứ không phải địa vị mà công ty giao cho bạn, hay do những dựa dẫm của cấp dưới. Nhưng quyền uy là nhu cầu về hình tượng quyền lực của mọi người, nhằm đảm bảo rằng có người quan tâm đến anh, bảo vệ lợi ích của anh, tôn trọng anh, chứ không phải uy quyền thực sự.

Vậy uy quyền thực sự có những ảnh hưởng gì? Đó là bạn biết mình đang làm gì, để thành viên hiểu bạn đang làm gì, để thành viên và bạn cùng nỗ lực tạo giá trị cho mục tiêu đã định.

Nhóm trong giai đoạn trẻ sẽ không biết cách tiếp nhận quyền được trao, có người lo lắng không có mục tiêu, thiếu mục tiêu rõ ràng, không tự nguyện làm việc khó.... Lúc đó, cách duy nhất để kích thích nhóm phát triển là tập trung sức lực vào hoàn thành sản phẩm đúng hạn. quản lý chỉ cần chịu trách nhiệm về tiến độ và chất lượng sản phẩm, phải bàn khoản về ranh giới phân công công việc và địa vị, chú tâm hoàn thành sản phẩm đúng hạn, đơn giản hoá mục tiêu, loại trừ mọi

phiền nhiễu. Quản lý dùng thái độ đó đưa vào công việc của nhóm, sẽ tự phá vỡ quan niệm bạo quyền, đưa nhóm tự phát triển. Đó chính là "uy quyền", là hành động, là cùng nhau xây dựng sản phẩm phần mềm, chứ không phải là ai quyết định cái gì, hay ai có quyền quyết định thưởng phạt.

CẠNH TRANH

Trong bất cứ thị trường nào, đều có một trong bốn tình huống sau:

- Không có đối thủ cạnh tranh.
- Không có phân thắng bại với đối thủ cạnh tranh.
- Tụt hậu hơn so với đối thủ cạnh tranh.
- Dẫn trước đối thủ cạnh tranh.

Dù là tình huống nào, đều phải có phán đoán chính xác về hướng đi của thị trường tương lai, động hướng của đối thủ, địa vị và hướng cạnh tranh của mình.

NHÂN LOẠI HỌC THU NHỎ

Trước khi phân tích tính chất bốn loại thị trường và đối sách thích hợp, chúng ta hãy bớt chút thời gian nghiên cứu kinh nghiệm cạnh tranh của thiên nhiên. Tôi nghĩ rằng để cho những nhà nhân loại học bàn sẽ hay hơn, nhưng tôi thấy về bản chất môi trường cạnh tranh trên thương trường cũng là thu nhỏ của cạnh tranh của con người, thậm chí còn khốc liệt hơn. Trong cạnh tranh, chỉ có hai góc độ là chiến thắng và thất bại.

Uy hiếp, mộng tưởng, sợ hãi và hy vọng đều nằm trong lòng những người tham gia, dưới sự tác động của nội tâm phức tạp với những tâm tư đối lập, kích thích tạo ra năng lượng, giúp người ta dốc hết tâm trí vào sáng tạo những thành quả to lớn.

Cạnh tranh thương mại bắt nguồn từ khát vọng nguyên thuỷ của con người với chiến tranh, bản chất chính là chinh phục kẻ thù, chiếm đoạt tài sản và phá huỷ tương lai của họ, so với người dã man thì chỉ đẹp hơn về hình thức mà thôi!

Cạnh tranh chính là chiếm đoạt tài nguyên của người khác làm của mình, thay thế môi trường sống của con cháu đối thủ thành môi trường sống tốt hơn cho con cháu mình.

Tuy cạnh tranh hiện nay của con người không còn là sát hại sinh mạng nhau, nhưng lại xuất hiện những cuộc cạnh tranh tài nguyên giữa các nhóm hơn nữa không phải với mục tiêu là thú săn, tài sản hay đất đai. Chúng ta có lẽ đều có một kết luận chung: Con người hiện nay, có xu hướng kết hợp từng nhóm để theo đuổi một mục tiêu nào đó, chúng ta gọi xu hướng hợp tác này là "*Hợp tác nhóm*" (teamwork). Đối với một số người danh từ này chỉ có nghĩa là hiền từ một cách yếu ớt hoặc ý tốt vô tác dụng, nhưng một ví dụ lớn hơn là: Một bầy sói hung ác đang ăn một con cừu, đàn sói sẽ

không vì tranh ăn mà tàn sát lẫn nhau, mà chỉ có thể giết con vật mà nó săn được.

Xâm lược nhóm có tính huỷ diệt, và có vẻ rất khó nghe. Trong thực tế hành vi cạnh tranh trên thương trường chỉ là phiên bản hiện đại trong cuộc sống của con người hàng ngàn năm nay. Chúng ta có thể suy đoán một cách hợp lý, sở dĩ con người lựa chọn khả năng hợp tác nhóm là vì đây là phương thức thành công rất hữu hiệu, có hiệu quả hơn bất cứ hợp tác nào không thuộc nhóm. Do đó chúng ta không thể không tin rằng cạnh tranh như vậy và quá trình diễn biến vẫn không ngừng xảy ra.

Tôi không muốn đưa ra bất cứ phán đoán đạo đức nào về đặc tính này của con người, nhưng tôi có khuynh hướng tin rằng đặc tính hợp tác nhóm này có vai trò rất quan trọng trong quá trình tiến hoá nhân loại. Văn minh, thấu hiểu, yêu, trung thành, tin cậy hay thậm chí xảo quyệt đều bắt nguồn từ đặc tính này. Mong muốn chinh phục đối thủ, săn mồi, hay giành giật không gian sinh tồn luôn xuất hiện trong hoạt động của con người, khi suy nghĩ đến tổ chức đoàn thể và tình hình cạnh tranh đều cần để ý đến các nhân tố này. Các thành viên trong nhóm của bạn, đều là những thế hệ đã từng hợp tác và giành được thành công.

Nếu chúng ta coi đối thủ cạnh tranh như kẻ thù, vậy thị trường chính là con mồi mà mọi người tranh giành, hoặc là mảnh đất màu mỡ cho doanh nghiệp sau này. Nếu chúng ta chiếm được thị trường, cũng chính là chiếm được tài sản của đối thủ, đuổi đối thủ đi hoặc tiêu diệt họ.

Chúng ta cần phải hiểu được, bất cứ thị trường nào có lợi đều xuất hiện đối thủ cạnh tranh, chỉ là vấn đề sớm hay muộn mà thôi. Đối thủ sẽ tổ chức nhóm của họ để tranh hơn với bạn, chiếm thị trường, tìm mọi cách để loại bạn ra khỏi thị trường, bạn cũng như vậy.

CẠNH TRANH PHẦN MỀM

Trước khi triển khai bất kỳ dự án nào, cần phải đánh giá chi tiết về hướng cạnh tranh. Trong ngành phần mềm rất khó sinh tồn này, cạnh tranh có tính chất cá nhân, không như các ngành truyền thống cả doanh nghiệp đều cạnh tranh trong lĩnh vực rộng. Phát triển phần mềm là suy nghĩ và sáng tạo cá nhân, mà cạnh tranh lại là giữa trí tuệ của các cá nhân, hướng cạnh tranh cũng thay đổi nhanh hơn bất kỳ một ngành truyền thống nào. Đối với mỗi đơn đặt hàng trong ngành truyền thống, đều so sánh xem ai xử lý nhanh hơn, ngành phần mềm thì phức tạp hơn nhiều. Chúng ta đã đứng ở đầu cùng của khoa học kỹ thuật thông tin,

một số sản phẩm phần mềm hay kỹ thuật chỉ trong một năm đã thay đổi hoàn toàn. Bởi chu kỳ sống của sản phẩm chưa hết, nhưng thị trường đã thay đổi căn bản, do đó rất khó có thể coi cạnh tranh phần mềm như một thị trường đơn lẻ.

Do chu kỳ sống của sản phẩm chưa hết, nhưng thị trường đã thay đổi căn bản, nên rất khó có thể coi cạnh tranh phần mềm như một thị trường đơn lẻ.

Do ngành phần mềm thay đổi quá nhanh, chiến lược kinh doanh có tính toàn cầu của sản phẩm phần mềm cần phải vừa mới mẻ vừa sâu sắc. Tiến bộ của khoa học kỹ thuật cao làm cho người ta tiến lên với tốc độ cao, mỗi ngày đều có thay đổi mới. Phiên bản năm 2000 đương nhiên có nhiều chức năng tốt hơn phiên bản năm 1999, và nhiệm vụ quan trọng mà cấp thiết là phải giới thiệu cho cả thế giới biết phiên bản mới có gì mới, có tính năng gì đáng giá.

Nếu bạn không thể luôn nắm được mạch đập của thời đại, nếu bạn mệt mỏi trong biến chuyển nhanh chóng và khốc liệt của môi trường xung quanh, nhất là những hành động của đối thủ, nếu bạn không thể tiếp tục sáng tạo các kỹ thuật vốn có, đảm bảo luôn dẫn đầu thì đối thủ sẽ thừa cơ tiến lên, thay đổi chỗ bạn và trở

thành người chiến thắng trên thị trường, chiếm lại được khách hàng.

Sau khi xác định được tình hình của bạn, cần suy nghĩ sử dụng các bước đi tiêu chuẩn

Rất rõ ràng, trong bất kỳ một môi trường cạnh tranh nào đều có rất nhiều cách để phản kích đối thủ, nhưng chúng ta chỉ thảo luận về bốn loại điển hình trên. Cũng như thế cờ vậy, có nhiều bước đi tiêu chuẩn mở màn, cũng như khi đối chọi, dùng cách tấn công gì, cách đối phó nào đều cần tuỳ thuộc vào tình hình của bạn.

NGUYÊN TẮC 10: KHÔNG CÓ ĐỐI THỦ CHƯA CHẮC ĐÃ LÀ ĐIỀU TỐT

Alone? A market without a competitor ain't

Chúng ta thường nghe nói đến một số công ty cho rằng mình không có đối thủ cạnh tranh trên thị trường, vì họ có sản phẩm đặc sắc, kênh tiêu thụ độc đáo, hoặc định vị của họ độc nhất vô nhị, không ai có thể cạnh tranh. Đương nhiên, trong một số trường hợp, cách nói trên có thể đúng, nhất là những thị trường mới, hoặc những thị trường rất chuyên môn, có bản chất riêng. Nhưng thông thường, cách định vị đó chưa hợp lý, bởi:

Trước hết, cách nói này đại diện một quan niệm giá trị sai lầm, cho rằng "không có đối thủ cạnh tranh" là điều tốt, có một số công ty kiêu hãnh nói rằng mình không có đối thủ, và cho rằng đó là hiện tượng đáng tự hào. Quan điểm này chưa hẳn đã đúng, nếu một thị trường đủ mạnh, nó sẽ thu hút đối thủ cạnh tranh. Trong thực tế, đối thủ thường sử dụng chiến lược tiếp thị có chức năng giống của bạn, "giúp" bạn tạo ra một thị trường, cùng bạn khai thác thị trường.

Thứ hai, nếu đối thủ theo sát không rời, sao chép những đặc trưng sản phẩm mà bạn thành công nhất, bạn sẽ thấy rất khó khăn dẫn dắt thị trường. Nhưng

động tác sao chép của đối thủ cũng có nghĩa là khẳng định địa vị dẫn đầu thị trường của bạn. Kinh nghiệm cho thấy, trở thành duy nhất trong thị trường sẽ dẫn đến nhiều thứ bất ổn, chưa phải là điều tốt. Nếu bạn quyết định đầu tư một dự án kỹ thuật mới vượt thời đại, bạn cần có một cái mốc để chứng minh rằng mình đã đủ tiến bộ, đồng thời tạo nên những tác động tới thị trường. Nếu không ai cần đến thị trường của bạn thì thị trường đó còn có giá trị gì nữa.

Thứ ba, nếu bạn là duy nhất trên thị trường, sẽ không thể xây dựng hình tượng độc đáo rõ rệt trong lòng khách hàng. Quả thật, trong lòng khách hàng công ty hay sản phẩm của bạn đều giống nhau cả, bởi họ không hiểu rõ về đặc điểm kỹ thuật của bạn. Khách hàng vừa không có lựa chọn, cũng không hề thừa nhận. Họ phải được so sánh rồi lựa chọn sản phẩm của bạn, họ mới phân biệt được đặc điểm của bạn, cũng mới có thể giải thích được tại sao họ chọn như vậy. Nhưng nếu bạn là duy nhất trên thị trường, khách hàng không cần thiết phải phân biệt đặc trưng của bạn, hay tìm hiểu điểm khác biệt của bạn, so với người khác. Có rất nhiều biện pháp tốt không được chấp nhận, vì đó vốn là biện pháp duy nhất, không thể có đối tượng so sánh để tìm ra sự khác biệt. Mỗi mục mua về đều mang tính mạo hiểm. Truy trong thời kỳ đầu của phần mềm, xưởng sản xuất có thể hợp tác chặt chẽ với khách hàng, càng nỗ

lực và tạo ra mối quan hệ tốt. Nhưng đa số các trường hợp đều là: Nếu khách hàng mua một sản phẩm duy nhất trên thị trường sôi động, họ sẽ dần cảm thấy thất vọng.

Độc nhất trên thị trường, rất khó có thể chuyển các thông tin độc đáo về sản phẩm cho khách hàng.

Đương nhiên, ở phía trước chúng ta đã nói qua, nếu là duy nhất trong thị trường sẽ có hai tình huống: Một là thị trường mới. Để dẫn đầu trong thị trường, nhanh chóng chiếm lĩnh thị phần là mong muốn của mỗi công ty phần mềm; Nhưng đáng buồn là, cần rất nhiều vốn đầu tư để khai thác trên một thị trường mới toanh, rủi ro lớn, chỉ những công ty có nguồn vốn lớn mới dám nghĩ đến, hoặc là những công ty sớm đã chiếm được khách hàng.

Khai thác thị trường mới tối thiểu có hai vấn đề: Trước hết, chuyển thông tin tới khách hàng như thế nào? Chức năng mới và kỹ thuật mới thường rất phức tạp, rất khó giải thích rõ ràng, cần phải trích dẫn và cổ động rất lớn mới có thể giúp khách hàng hiểu rõ. Trừ phi bạn có một chuyên gia giúp mọi người thấu hiểu, đồng thời đúng lúc khách hàng rất cần, mà sản phẩm

của bạn lại trong tâm điểm, nếu không chi phí tiếp thị sẽ không tỷ lệ với hiệu quả đem lại.

Vấn đề thứ hai là, sản phẩm mới thường cần hoạt động tiếp thị rất lớn, thời gian để càng lâu, tỷ lệ thắng càng giảm. Sau khi bạn đóng gói, xây dựng hình tượng sản phẩm, định giá, xây dựng kênh phân phối và quảng bá thông thường thì bạn cũng không còn bao nhiêu vốn liếng nữa, có khi còn không đợi đến lúc có hợp đồng thứ hai.

Đối với cạnh tranh độc nhất này, không cần phải nói nhiều hơn, nếu bạn không có đối thủ, thì bạn phải biết và hiểu cách duy trì thị trường của mình.

NGUYÊN TẮC 11: ĐỐI THỦ ĐUỔI SÁT? TUNG RA NHỮNG TÍNH NĂNG MỚI

Dead beat? Break out of a feature shoot-out

Trong xu thế thị trường cạnh tranh khốc liệt, nhất là khi chưa có ai nổi trội, mọi người đều tranh tung ra những tính năng sản phẩm mới, để tỏ vẻ khác biệt với đối thủ và thu hút sự chú ý của khách hàng. Lúc này trọng điểm đánh giá ưu nhược điểm của sản phẩm là xem sản phẩm có những tính năng gì, cũng như cấp độ chất lượng. Bạn đã nghe về *hộp kiểm* (check box, hộp chọn, dùng để khách hàng đánh dấu nếu chọn lựa), những sản phẩm có tính năng như vậy thường dùng để so sánh, nhận xét hay lựa chọn, xem xem các sản phẩm trên thị trường, sản phẩm nào có (hay không có) tính năng này. Kết quả là phải tốn thêm thời gian xây dựng tính năng này cho một số khách hàng cần lựa chọn, hoặc do một số nhu cầu trong kinh doanh, giúp họ nhận thấy sản phẩm này không bị lạc hậu.

Giá thành cho phương thức cạnh tranh này rất cao, đồng thời cũng không hiệu quả.

Càng có nhiều tính năng, càng làm cho sản phẩm phình to và yếu hơn.

Nếu bỏ qua các nhân tố định giá, kênh tiêu thụ, dịch vụ, ... mà chỉ xét bản thân phần mềm, có hai chiến lược chính để giành chiến thắng về tính năng sản phẩm. Phương thức thứ nhất là bạn đưa ra tính năng sản phẩm luôn vượt trước đối thủ, làm họ không thể đuổi kịp trong một hai chu kỳ sản phẩm. Khó khăn của chiến lược này là, do sáng tạo của bạn cần phải có bước tiến dài, sự phức tạp của sản phẩm tăng lên rõ rệt, rất khó có thể dùng phương pháp đơn giản mà hiệu quả để hướng dẫn khách hàng sử dụng như thế nào, có những lợi ích gì, ... Hơn nữa tính năng sẽ ngày một khó làm hơn, sản phẩm mềm sẽ phình to và yếu đi, duy trì tính ổn định rất khó khăn, sản phẩm càng lớn, thời gian để nâng cấp phiên bản mới càng lâu, đây là điều khó tránh khỏi.

Khi bạn dùng chiến lược cạnh tranh này, cần thiết là đối thủ cũng sử dụng chiến lược này và bạn phải xác định được tính năng sản phẩm của đối thủ không thể so với của bạn, không thể đuổi kịp trong một thời gian ngắn. Phương pháp này đáng để thử, nhưng không phải là phương pháp tốt nhất, hơn nữa làm như vậy không thể xây dựng được phần mềm lớn.

Một chiến lược nữa để giành chiến thắng trong cạnh tranh tính năng sản phẩm là đầu tư lớn vào tính năng sản phẩm mẫu mực, hơn nữa tính năng sản phẩm

phải nhiều đến mức đối thủ không thể so sánh với bạn trong lĩnh vực có tính mẫu mực này, tối thiểu phải vượt ba tính năng mới đủ an toàn. Một khi bạn đã xưng hùng được thì đối thủ phải tạm rút lui, dù họ có vẻ hơi nổi trội hơn. Điều quan trọng là, có thể thực sự thay đổi phương thức làm việc của người dùng, thì có thể thắng lợi gấp nhiều lần, nếu không chỉ là tô điểm cho các tính năng mà thôi.

NGUYÊN TẮC 12: TỤT HẬU? TĂNG CƯỜNG ĐẦU TƯ, NHANH CHÓNG TUNG RA PHIÊN BẢN MỚI

Behind? Ship more often with new stuffs

Trong ngành phần mềm không thể có hai thất bại hoàn toàn giống nhau. Sản phẩm thất bại có rất nhiều nhân tố không rõ ràng, trong đó thường gặp nhất là phiên bản mới không so được với sản phẩm của đối thủ. Nếu đối thủ vượt bạn một vài chu kỳ của phiên bản thì bạn đang trong tình thế nguy hiểm, không thể không kiểm điểm triệt để và tái định vị.

Nếu bạn gặp may, còn sống sót thì bạn vẫn có cơ hội phấn đấu giành chiến thắng.

Tập trung sức lực nâng cao sản phẩm vốn cần phải đầu tư nhiều, hơn nữa trong một thời gian ngắn phải xây dựng nhiều tính năng mới sẽ làm loạn cả nhóm, kết quả sẽ còn nguy hiểm hơn, đủ để loại bất cứ sản phẩm phần mềm nào khỏi thị trường. Do tụt hậu, ý chí của nhóm đã suy thoái rất nhiều. Liều thuốc mạnh này có thể còn nguy hiểm hơn, cũng như điều trị hoá

chất vậy, chữa trị có thể gây thương tổn lớn hơn bản thân căn bệnh đó.

Phát hiện mình tụt hậu sẽ làm cho người ta đau buồn, giới truyền thông, cấp trên, đồng nghiệp, khách hàng, cả thế giới đều biết bạn có biểu hiện kém. Đó là cảm giác rất đáng sợ, hơn nữa kết quả cũng không biết sẽ ra sao.

Nếu bạn may mắn còn sống sót, bạn vẫn còn cơ hội để phấn đấu. Nhưng buồn phiền vẫn quấy nhiễu xung quanh. Nhất là khi bạn phải cho cả thế giới biết quyết tâm bại không nản của mình. Điều này rất quan trọng, vì không ai mua sản phẩm mà công ty phát triển phần mềm sắp vứt đi.

Giới truyền thông đánh giá khả năng thì cứ mặc họ phê bình. Dù sao thì có biện hộ thế nào họ vẫn nhận xét như thế. Chỉ bằng chuyên tâm tập trung tất cả tài nguyên, chuẩn bị kỹ cho trận chiến tới, bạn chỉ thua một lần chứ không phải đánh mất tất cả. Bảo toàn thực lực, bạn vẫn còn cơ hội để chiến thắng.

Không cần tuyên bố kế hoạch trước. Mặc dù bạn cần phải tuyên truyền, cần phải lấy lại thể diện, cũng cần cho khách hàng cũ một tia hy vọng, nhưng không được buông lời trước khi hoàn thành sản phẩm. Làm như vậy rất nguy hiểm, chẳng khác gì giảm hiệu ứng khi tung sản phẩm ra.

Đối với bản thân và nội bộ nhóm đều phải thừa nhận sự thực thất bại, không được giả như không có chuyện gì (Không cần nói cho người khác biết). Thực sự tìm ra lỗi quan trọng nhất, sửa đổi thực sự và sau đó lại bắt đầu.

Nếu vấn đề là lỗi phần mềm quá nghiêm trọng, đương nhiên phải dọn sạch trong phiên bản tới, tối thiểu phải loại được các lỗi chính. Đây là công việc tốn rất nhiều thời gian, có thể lấy sạch tài nguyên của bạn. Nếu có như vậy, cũng không được thu phí từ bản sửa lỗi đó bởi lỗi đó không phải do khách hàng. Tôi cho rằng chiếm được vị thế trên thị trường cần đến một từ: Tàn nhẫn. Bạn phải tàn nhẫn khi tìm kiếm sự vượt trội, cũng như tàn nhẫn khi tấn công. Chiến thuật hữu hiệu nhất là đưa ra sản phẩm mới nhanh hơn đối thủ. Quan niệm này chính là: Làm cho khách hàng tin tưởng vào bạn, tỏ rõ ý đồ của bạn. Nếu cạnh tranh phần mềm như một trận đấu, khách hàng chính là khán giả. Khán giả sẽ thích đội suất sắc, và họ có quyền quyết định ai thắng ai thua. Do đó, hãy đứng dậy! Dù cục diện không có lợi, cho khách hàng thấy rõ thực lực của bạn, xem cách giành chiến thắng của bạn.

Nếu nhóm của bạn có thể đưa ra sản phẩm mới nhanh hơn đối thủ, đồng thời vốn đầu tư được hạn chế ở

mức độ hợp lý, cuối cùng bạn sẽ thắng! Tung sản phẩm ra là vấn đề khó khăn nhất, nếu bạn làm tốt hơn đối thủ, bạn sẽ có cơ hội để biểu hiện các điểm tốt hơn khác. Đưa ra sản phẩm mới đúng hạn và thường xuyên là bảo bối lớn nhất trong ngành phát triển phần mềm.

NGUYÊN TẮC 13: DẪN ĐẦU? KHÔNG ĐƯỢC QUAY LẠI

Ahead? Don't ever look back

Bạn đã dẫn đầu? Chúc mừng bạn cuối cùng đã xưng hùng xưng bá, khó khăn hiện nay của bạn là tiếp tục dẫn đầu hướng đi của sản phẩm trong lĩnh vực đó, mở ra kỹ thuật mới và các cơ hội mới đi kèm. Tự mãn là kẻ thù lớn nhất của bạn, hãy ghi nhớ mấy việc sau:

Bạn không thể duy trì việc luôn dẫn đầu. Luôn có người cản phá bạn bất cứ lúc nào, bạn không biết đối thủ là ai? Dùng phương pháp gì? Khi nào?

Bạn phải cạnh tranh với chính mình. Bây giờ bạn phải tăng đầu tư vào tính năng có tính mẫu mực gấp 3 lần, nội dung mỗi lần nâng cấp phải phong phú hơn (dù bạn đã giữ chức quán quân), tạo dựng địa vị cao hơn cho bạn.

Cho tất cả mọi người thấy là bạn đang tiến mạnh về phía trước, không hề quay đầu lại.

Hãy nâng tốc độ thay đổi đến mức không ai có thể đuổi kịp. Là người dẫn dắt thị trường, bạn có cơ hội tự

quyết định lịch trình và tốc độ nâng cấp. Về lý mà nói, lợi ích mà bạn thu được cao hơn người khác, bạn phải tăng đầu tư để ổn định địa vị dẫn đầu này. Giành được thị trường rất khó, nhưng duy trì nó còn khó hơn nhiều. Hãy nhớ rằng, nếu đối thủ nắm được quyền khống chế tốc độ nâng cấp, có thể bạn sẽ mất vị trí dẫn đầu, tối thiểu cũng bị ép phải đưa ra phản ứng tương ứng. Chu kỳ sống của sản phẩm khoa học kỹ thuật cao được tính theo tháng, chứ không phải theo năm, mà sự thay thế của các thế hệ phần mềm diễn ra rất nhanh, có thể biến chuyển bất cứ lúc nào, do đó bạn phải luôn giữ được khả năng hành động nhạy bén của mình.

Cần hạ quyết tâm làm đến cùng. Ngoài việc củng cố địa vị hiện có, bạn còn phải tiếp tục thách thức những mục tiêu cao hơn. Hãy cho mọi người biết rằng bạn đang tiến bước, không hề quay lại. Bạn không được để hiện thực hạn chế bước đi của mình, tính tương thích rất quan trọng, khi tiến lên phía trước bạn phải đảm bảo sản phẩm có tính tương thích, để khách hàng có thể đi cùng bạn, chứ không được để cho khách hàng hay ai đó giữ lại.

NGUYÊN TẮC 14: ĐẢM BẢO SỰ MỚI MẺ

Take the oxygen along

Khi bạn quyết định phát triển phần mềm máy tính cá nhân, nhất là khi dùng trên môi trường Windows, bạn cần phải có nhận thức: Bạn không chỉ phát triển phần mềm, mà bạn đang lựa chọn một kiểu triết học sống. Tốc độ thay đổi của nó diễn ra chóng mặt, cũng giống như sự biến thiên của khoa học kỹ thuật và xã hội. Nhanh đến mức bạn chưa kịp thích ứng đã phải nâng cấp. Quá trình phân tích, phát triển, xây dựng, duy trì không còn là một công thức được thao tác hoá nữa.

Hệ điều hành mang tính toàn cầu liên tục nâng cấp đã trở thành đặc tính cơ bản của thời đại thông tin, thúc đẩy phần mềm không ngừng thay đổi. Không chỉ là phần mềm, mà cả phần cứng, các thiết bị ngoại vi và các hãng thông tin đại chúng đều bị ảnh hưởng lớn do hệ điều hành nâng cấp. Tiết tấu biến đổi nhanh chóng này là trạng thái thông thường của xã hội thông tin, cũng là thứ bạn không thể đối kháng được.

Tiết tấu biến đổi nhanh chóng là trạng thái thông thường trong xã hội thông tin, bạn phải tiến lên, nếu không sẽ bị tụt hậu.

Có phản ứng nhanh nhạy với khoa học kỹ thuật thông tin toàn cầu là điều kiện cơ bản của tổ chức lành mạnh. Xưởng đóng gói phần mềm đóng gói kỹ thuật của họ vào phần mềm để bán cho khách hàng, mà bản thân sản phẩm là công cụ chính để kết nối mọi người. Để giải quyết vấn đề hiện tại, xưởng phát triển phiên bản nâng cấp, đó là duy trì hành vi mua hàng có tính liên tục, chứ không phải mua một lần rồi thôi.

Xưởng hệ điều hành không ngừng đưa ra bản nâng cấp, không chỉ vì lợi ích của mình (Tuy làm như vậy sẽ có lợi cho họ), mà mục đích chính là đưa khoa học kỹ thuật mới vào trong sản phẩm, để cho khách hàng sử dụng. Nếu hãng phát triển phần mềm ứng dụng trên máy tính cá nhân bỏ qua một phiên bản của hệ điều hành, khách hàng sẽ không thể sử dụng những điểm hay của hệ điều hành mới, chẳng hạn không thể tăng tốc độ hoặc cảm thấy đang dùng sản phẩm lỗi thời.

Không được coi thường sức mạnh của trào lưu. Hãy thử xem mong muốn và cảm nhận của bạn đối với xe máy, thời trang hay âm nhạc, sẽ hiểu rõ rằng không thể ngăn cản trào lưu. Khách hàng cũng thích khoa học kỹ thuật mới nhất, tiện lợi nhất. Do vậy, hãy đưa trào lưu vào trong kế hoạch phát triển sản phẩm của bạn!

KHÁCH HÀNG

Do đa số các hành vi mua phần mềm đều là mua bán nâng cấp cho phần mềm hiện có, nên các công ty phần mềm đều cần duy trì quan hệ lâu dài tốt đẹp với khách hàng, nhưng đây không phải là vấn đề dễ dàng. Chúng tôi không bàn đến hành vi mua hàng chỉ một lần là thôi. Về cơ bản, sống trong thế giới khoa học kỹ thuật tiến bộ, khách hàng đều mong muốn nâng cấp phần mềm. "Quan hệ duy trì" giữa khách hàng - hãng cung cấp này rất mật thiết và phức tạp đồng thời cũng rất quan trọng trong ngành phần mềm.

Sở dĩ mối quan hệ khách hàng - hãng cung cấp phức tạp đến kinh ngạc là do khách hàng mất rất nhiều thời gian sử dụng phần mềm, và nghiên cứu khá sâu. Lấy ngay Visual C++ của chúng tôi làm ví dụ, tôi đảm bảo khách hàng mỗi ngày dùng nó hơn 8 tiếng, thời gian sống trong thế giới ảo này còn dài hơn thời gian trong thế giới thực. Thời gian dùng sản phẩm còn nhiều hơn thời gian làm việc hay nói chuyện với mọi người, đây là việc đáng phải chú ý!

Nguyên nhân thứ hai làm cho mối quan hệ khách hàng - hãng cung cấp trở nên phức tạp là: Phần mềm không chỉ tiêu tốn thời gian của họ, mà còn gây hạn chế cho tiềm năng của họ. Để có thể trở nên tự động hoá và sử dụng hiệu quả, cần phải đưa ra rất nhiều điều kiện

hạn chế cho phần mềm, nhưng đồng thời cũng hạn chế luôn biểu hiện tự do của người dùng.

KHÁCH HÀNG BỊ PHẦN MỀM TRÓI BUỘC.

Nếu khách hàng thích sản phẩm của bạn, cần phải chú ý đến những thảo luận dưới đây. Khách hàng lựa chọn phần mềm của bạn là do nhân tố tâm lý và tâm tư, chứ không phải nhu cầu làm việc. Khi mua sản phẩm của bạn, cũng như mua các mặt hàng khác, đều có niềm tin nhất định. Lúc này bạn phải thật cẩn thận, mối quan hệ khách hàng - hãng cung cấp rất dễ có vấn đề.

Khách hàng chọn bạn không theo lý tính, cũng chưa hề so sánh với các sản phẩm khác. Khi đã hết nhiệt tình, họ sẽ cảm thấy chán ghét, đồng thời cảm nhận về sản phẩm của bạn một cách không công bằng. Bạn phải bù đắp những ân hận của họ do không chọn lựa khi mua; Bạn phải làm cho khách hàng tình nguyện chọn bạn. Cho dù hiện nay bị phần mềm của bạn trói buộc, nhưng không phải đã lên nhầm thuyền mà đành phải chọn bạn. Lúc này, bạn ở trong trạng thái phản diện, cần làm cho khách hàng cảm thấy họ đã không quyết định nhầm.

Mọi động tác của phần mềm đều qua thiết kế (Ít nhiều đều hỏi ý kiến chuyên gia). Người thiết kế tìm ra chương trình thông thường nhất, quy nạp một nhóm

động tác tiêu chuẩn, sau đó lựa chọn một số chức năng tự động, quy định các quy tắc nhập hay xuất, cần phải chạy trên máy nào, dùng phần mềm nào để thực hiện. Mục tiêu người dùng của người thiết kế là đa số những người bình thường, người thiết kế mong rằng có thể làm cho càng nhiều người được thoả mãn nhiều nhất. Cách làm này không sai, nhưng làm cho người ta cảm thấy bị cản trở.

Tuy người dùng mất nhiều thời gian cho phần mềm, nhưng lại không có ảnh hưởng đến thiết kế. Phát triển phần mềm như động tác của hộp đen, có rất nhiều thông tin được đưa vào, các lập trình viên sẽ tìm ra loại hình chiếm đa số, suy nghĩ đến nhu cầu chứ không nói trực tiếp với người dùng. Sản phẩm như vậy dùng để phục vụ thị trường, chứ không phục vụ khách hàng, cách làm này trên thực tế đã lấy đi rất nhiều tiềm lực của con người.

Sự không cân đối này là cần thiết, nhưng lại làm phức tạp mối quan hệ với khách hàng. Đúng vậy, ai cũng có thể nói, công ty phần mềm không phát triển đủ các công cụ (Có thể do nhân tố kỹ thuật hay thời gian), mà chỉ chú tâm phát triển các phần mềm ứng dụng cho riêng mình, công ty phần mềm thường làm các phần mềm mà họ cho là điển hình, mà không cho khách hàng sự tự do khi sử dụng. Tuy mỗi người đều mong muốn

làm việc theo phương thức của mình, nhưng đa số phần mềm đều yêu cầu người dùng phải sử dụng phương thức đặc biệt để thoả mãn một chút cá tính riêng. Vậy thôi, chỉ cần làm được, khách hàng sẽ miễn cưỡng thấy vừa ý. Hãng phần mềm ứng dụng có thể tăng nỗ lực về điểm này, để cho khách hàng có thêm không gian cho riêng mình.

Không có nó làm gì cũng không xong, nhưng khi có rồi tôi vẫn chưa làm được điều mình muốn.

Nhưng do sự khác biệt của mỗi người mà cần phải cho thêm nhiều đặc trưng vào công cụ của phần mềm, ngược lại có thể làm cho phần mềm trở nên phức tạp và không được mọi người chấp nhận. Sự phức tạp không thể chấp nhận chưa chắc do các đặc trưng gây ra, nhưng chúng thường đi liền nhau. Do rất khó thiết lập các đặc trưng riêng, mong muốn tự thiết lập hay bị thất bại. Nói cụ thể hơn, chính là đưa ra kết quả có tính đàn hồi, sẽ vô tình hay cố ý làm hại mọi người gặp vấn đề, sau đó làm rối tung lên, ai cũng không hài lòng. Về bản chất giữa "Dễ sử dụng" và "Tiện thay đổi" tồn tại xung đột và đối lập. Nếu bạn giảm thiểu vấn đề này, sẽ đáp ứng được càng nhiều nhu cầu khác nhau của khách hàng. Để phần mềm dễ sử dụng là cả một học vấn lớn, còn làm phần mềm có thể vận dụng linh hoạt càng khó

hơn. Nếu bạn làm cho chúng đơn giản hơn, sẽ rất dễ thu hút khách hàng "Thay đổi nhu cầu chính là yêu cầu của tôi".

Trong thế giới có thời gian và tài nguyên hữu hạn, hãng phần mềm cần phải chọn tính năng thích hợp để phát triển, đương nhiên như vậy sẽ hạn chế không gian sử dụng của người dùng. Càng ngày càng nhiều người dùng chán nản về hạn chế của phần mềm, có thể đáp ứng nhu cầu công việc hàng ngày, nhưng không thể nói là vừa ý. Người dùng phần mềm cuối cùng rất buồn bã với hiện thực, ngày nào cũng dùng, những hạn chế rất khó chịu, nhưng không có chúng lại không ổn. Cũng như mỗi lần đi giày đều cần dùng đón gót để đưa chân vào - nên cũng có thể coi đây là hiện tượng "đót gót".

Tôi thường hay diễn giảng về chủ đề phát triển phần mềm, mỗi khi bàn về quan hệ phần mềm với khách hàng đều hay dùng máy chiếu, phía trên viết "Hầu hết các phần mềm đều hỏng" (Most Software Sucks), lần nào cũng làm cho mọi người lăn ra cười và vỗ tay không ngớt. Tôi không thể không thừa nhận, phần mềm hiện nay tuy làm cho người ta chưa vừa lòng, nhưng vẫn còn tốt, do đó mới có thể lãi hàng chục tỷ đồng!

Tôi mua phần mềm, cũng rất thích chúng (lúc mới đầu) chỉ vì chúng rất tốt. Nhưng càng dùng càng thấy

bị hạn chế, còn không vừa ý bằng tôi tự viết. Tôi thiết lập lại cấu hình, vẫn không ổn, thế là tôi quyết định dùng phần mềm khác, kết quả là thấy mọi việc đều thay đổi thật, nhưng phần mềm mới còn tệ hại hơn, tôi đành quay về sử dụng phần mềm cũ. Tôi thật tức chết vì mấy cái này! Các hãng phần mềm thân mến, hãy cho tôi một cơ hội sử dụng phần mềm nâng cấp, không có nó tôi chẳng làm gì được cả, nhưng có nó rồi tôi vẫn chưa làm được thứ mình muốn.

PHƯƠNG THỨC MUA CỦA KHÁCH HÀNG

Xét trên góc độ khách hàng, quá trình tâm lý mua phần mềm đơn giản chia thành mấy giai đoạn, với các thứ tự: Trước hết, họ nhận thông tin về hãng, có *chú ý* (Attentive) tới sản phẩm, sau đó bắt đầu có *hứng thú* (Interested), muốn tìm hiểu về nó, rồi *tin tưởng* (Convinced) rằng mua nó là quyết định đúng đắn. Đương nhiên, nếu ngay từ đầu anh ta đã không thèm nghe thì sẽ không có những thay đổi tâm lý sau đó. Nhưng dù khách hàng đã có tâm lý tin tưởng, cũng chưa nhất định là sẽ mua, mà còn phải có *mong muốn* (Desire) về sản phẩm, thì họ mới thực hiện hành vi mua.

Bạn muốn làm cho khách hàng mong muốn về sản phẩm, thì phải suy nghĩ đến vấn đề này ngay từ

khi thiết kế. Tưởng tượng xem phần mềm của bạn có *thuộc tính* (Properties) gì có thể đáp ứng nhu cầu của khách hàng, cụ thể hơn, bản thân bạn phải "mong muốn" nó và biểu hiện cụ thể ra. Các chức năng khác có thể bàn bạc thêm, nhưng phải nhấn mạnh trọng tâm này. Suy nghĩ đi suy nghĩ lại về các thuộc tính, nghĩ lại các phiên bản trước xem đã làm đến mức độ nào về thuộc tính này, tìm cách để nó đi vào từng ngõ ngách chương trình, phải làm cho khách hàng thực sự thấy rõ thuộc tính này mới thôi. Phải làm cho bất cứ ai khi mới sử dụng đều hiểu nó và nói được ra thuộc tính này là gì?

MỸ HỌC PHẦN MỀM

Ở đây chúng ta hơi đi xa chủ đề một chút, để bàn về MỸ HỌC. Đa số mọi người đều nghe qua, nhưng chỉ có khái niệm rất mơ hồ, cảm thấy đó là kiến thức liên quan tới nghệ thuật hay thiết kế. Có lẽ bàn về MỸ HỌC từ góc độ phản diện sẽ dễ nắm được tinh thần của nó hơn, những thứ làm mọi người chán nản hoặc buồn ngủ không thể gọi là đẹp. Do đó, vẻ đẹp sẽ làm mọi người phấn chấn, nhạy cảm hoặc thoải mái.

Cái gì làm cho người ta phấn chấn? Đáp án là những vật dễ cảm nhận như hình dạng, màu sắc, âm thanh, động tác, cảm giác hoà hợp... Những thuộc tính này gây sự chú ý cho mọi người. Khi thiết kế sản phẩm,

đưa cảm giác đẹp vào sẽ làm cho người dùng vui vẻ, dễ chịu khi sử dụng.

Tác dụng phấn chấn tuy chỉ là phương tiện tâm lý, nhưng có thể ảnh hưởng đến hành vi khách hàng. Làm cho sản phẩm phần mềm của bạn có cảm giác đẹp, sẽ càng có giá trị sử dụng và tiềm năng thị trường.

NGUYÊN TẮC 15: LÀM KHÁCH HÀNG VUI MỪNG

Enrapture the customer

Đa số khách hàng đều mua phiên bản mới cho phần mềm sẵn có, sau một thời gian dài sử dụng, họ biết được hết ưu nhược điểm của phần mềm. Hơn nữa, do bộ phận kinh doanh không ngừng tuyên truyền sản phẩm và dịch vụ, thị trường hiểu rất rõ về công ty của bạn. Tóm lại, khách hàng biết về bạn.

Nếu khách hàng phải dựa vào phần mềm mà họ chưa được thoả mãn, họ sẽ càng cảm thấy không thoải mái, phần mềm họ dùng hàng ngày luôn trong trạng thái "đón gót", họ mong rằng bạn có thể sửa được trong phiên bản tới, chỉ cần giày rộng ra một chút cũng được.

Để hoà hợp với khách hàng cần đáp ứng được mong mỏi của họ, hiểu rõ điểm không vừa ý của họ trong mỗi phiên bản. Nếu bạn trung thực với khách, mong muốn nghe họ, tiếp thu ý kiến của họ, họ sẽ nguyện đi cùng bạn (Tuy có khi họ vẫn còn bực dọc trong lòng). Ví dụ, nếu trong hệ thống của bạn có một chương trình nhỏ rất phức tạp, làm cho khách hàng đau đầu, bạn sẽ phải cải tiến trong phiên bản tới. Hy vọng thấp nhất của khách hàng là bạn có thể hiểu được

những kinh nghiệm đau thương mà họ đã gặp, đồng thời biểu hiện điều đó trong phiên bản tới.

Mong muốn thấp nhất của khách hàng là bạn hiểu được những kinh nghiệm đau thương mà họ đã gặp phải.

Điều vừa kể trên là thành công thông thường, các sản phẩm lớn không như vậy, mà cần phải đưa trục kỹ thuật vào nhu cầu sâu kín nhất của khách hàng, chứ không phải mong muốn thấp nhất. Bạn phải dùng biện pháp sáng tạo, đáp ứng những nhu cầu khó nói ra của khách, dùng sản phẩm để cho thấy bạn hiểu rõ những ý nghĩa sâu kín, làm cho họ vui mừng và cảm động, loại bỏ những khuyết điểm làm người ta phẫn nộ, làm phần mềm phù hợp với nhu cầu của họ

Phần mềm lớn sẽ được chứng minh trên thị trường. Các phiên bản lớn và hoàn thiện không ngừng ra đời, cho khách hàng thấy bạn hiểu họ như thế nào, và thu hút rất nhiều khách hàng. Bạn sẽ được khách hàng coi như nguồn gốc của sức mạnh, do đó khách hàng cũng rất vui thích.

Vấn đề tự nhiên lại lớn, làm sao để biết cái gì làm phiền khách hàng. Điều quan trọng là, làm cách nào để dò xét nội tâm của họ? Câu trả lời thật đơn giản, trực

tiếp hỏi họ vào bất cứ lúc nào bất cứ nơi nào, bất cứ hình thức gì và qua bất cứ kênh nào.

Điều tra thị trường và các con số thống kê sẽ cho bạn biết thêm về thị trường, giúp bạn quyết định hướng sản phẩm tương lai, cho bạn những dữ liệu rõ ràng. Nhưng nếu bạn muốn xây dựng mối quan hệ tốt với khách hàng, thì đó là công việc có tính nghệ thuật chứ không có tính khoa học, hơn nữa bạn muốn biết các vấn đề quan trọng, vậy thảo luận dưới đây sẽ cho bạn vài phương thức khá đơn giản để hiểu khách hàng.

Gửi thư cho khách hàng hỏi họ thích hay không thích cái gì; Gọi điện cho họ; Trò chuyện trong các cuộc triển lãm; Tổ chức "Câu lạc bộ người tiêu dùng". Không có phương pháp nào là phức tạp hay tốn tiền. Câu lạc bộ người tiêu dùng chỉ cần có một địa điểm, mời một số khách hàng không xa lắm là có thể ngồi bàn rất nhiều chuyện, bạn chỉ cần chỉnh lý qua sổ ghi chép là sẽ có những ý kiến sâu sắc. Điều tra thị trường, nghiên cứu xem chúng ta cần phải biết vấn đề gì, dùng bảng hỏi đáp để điều tra sẽ có được những ý kiến rộng rãi.

Bạn không cần phải làm y hệt như trên. Trong nhóm phát triển phần mềm của thế giới hiện thực, nhất định sẽ tranh luận không ngừng về ý kiến khách hàng (Ý nghĩa thực chất mà khách hàng ám chỉ là gì, đưa ý kiến đó thực hiện vào phần mềm như thế nào,...). Điều

này rất bình thường, đồng thời cũng là hiện tượng tốt, có thể khuyến khích họ làm như vậy. Bạn có thể tiếp tục truy tìm nhu cầu của khách, ghi nhớ rằng thị trường ngày càng phức tạp, tìm hiểu nhu cầu khách hàng là công việc có tính liên tục, không phải là nhiệm vụ ngắn.

Để hiểu rõ nhu cầu của khách hàng đối với phần mềm cần phải có kỹ thuật, sự sáng tạo và không ngừng nỗ lực. Bạn phải nhận biết được nhu cầu trọng tâm của thị trường, tập trung kỹ thuật và tài nguyên để đáp ứng. Hiểu được nhu cầu trọng tâm sẽ có các hiệu quả sau:

- Sản phẩm đem lại cảm giác an toàn cho khách hàng.

- Sản phẩm giúp khách hàng nắm vững nhiều hơn.

- Nếu sản phẩm không bằng của đối thủ trong một số phương diện nào đó, nhưng vẫn đáp ứng nhu cầu trọng tâm của khách hàng, sản phẩm của bạn vẫn có hy vọng chiến thắng.

- Thông tin phản hồi của sản phẩm rất rõ ràng.

- Sản phẩm đơn giản dễ học.

NGUYÊN TẮC 16: TÌM KIẾM HỒNG TÂM

Find the sweet spot

Tôi tin rằng thị trường cũng có hồng tâm, hồng tâm chính là tập trung giá trị quan của mọi người. Hồng tâm luôn di động theo sự trưởng thành của thị trường, có lúc còn di chuyển rất nhanh. Nếu sản phẩm của bạn có thể nắm đúng trọng điểm mà đa số mọi người đều nhận định, tức là đúng hồng tâm, do vậy chỉ cần công bố hợp lý là sẽ được thị trường chấp nhận.

Khi chúng tôi phát triển Visual C++1.0, mọi người đều cho rằng C++ nhất định sẽ trở thành chủ chốt trên thị trường. Do các đại lý đều bày bán cả C và C++ nên không thể thống kê được mức độ tiếp nhận C++ trên thị trường qua các con số, chúng tôi chỉ có thể giả thiết, do trong nhóm sản phẩm này chủ yếu là nhu cầu về trình biên dịch C++ nên số lượng hàng bán đã chứng minh C++ được rất nhiều khách hàng chấp nhận.

Điều rất mừng là, khi đó các nhà phân tích, giới truyền thông và khách hàng đều cho chúng tôi biết rằng, thị trường mong muốn chúng tôi đưa *mẫu* (Template) và xử lý khác (Exception handling) vào trong sản phẩm. Đối với C++ mà nói, lúc đó vẫn khó thiết kế như vậy, đồng thời do hạn chế về thời gian, hầu

như không có cách nào để đáp ứng nhu cầu đó, hơn nữa chúng tôi nghi ngờ không biết khách hàng có thực sự cần chúng tôi làm điều đó không. Tôi không ngừng tự hỏi: "Tại sao phải đưa các tính năng phức tạp đó vào trong sản phẩm, và tôi dám cược là rất ít người có thể sử dụng chức năng phức tạp này". Trên thực tế đây chính là cơ sở lý luận trong kinh doanh, đi tìm cái độc đáo mới thoát khỏi sự trói buộc của truyền thống.

Trong khi thu thập thông tin thị trường, nhận được những chỗ kỳ lạ hoặc bất hợp lý, đi vào chính điểm này, thoát khỏi sự trói buộc của truyền thống.

Chúng tôi nghiên cứu và kết quả cho thấy chưa đến 15% người chuyển từ C sang C^{++}, tuy có 80% dự định, hoặc mong muốn sẽ chuyển sang vào năm sau. Do đó, chúng tôi bắt đầu tìm hiểu trực tiếp đối với khách hàng, nhằm tìm hiểu nhu cầu thực sự của họ. Thường khách hàng không cho bạn biết họ đang cần gì, nhất là khi đáp án đi ngược với truyền thống. Bởi họ không có cảm giác an toàn, họ có thể cho bạn biết họ cho rằng họ cần. Đó cũng là một trong những lý luận của kinh doanh: Khách hàng sợ khoa học kỹ thuật mới, nhân viên phát triển cũng vậy.

Nếu bạn bước vào phòng phát triển phần mềm và hỏi họ: "Có bao nhiêu người không biết dùng C^{++}?" Tôi nghĩ rằng chắc chắn sẽ không có ai đứng dậy nói: "Ồ. Chúng tôi còn đang chờ mẫu cơ", hay những lý do khác. Vì vậy, bạn cần phải tìm hiểu sâu về nhu cầu thực tế của họ, chứ không phải tính năng kỹ thuật đơn thuần này.

Cách làm thực sự hữu hiệu, là hỏi riêng anh ta: "Có phải anh gặp rắc rối khi dùng C^{++}?"

Lúc này có thể khách hàng nói thực lòng: "Đúng vậy, tôi thực sự không có thời gian học C^{++}, nó rất phức tạp, tôi xin đầu hàng. Ngay Windows tôi còn không nắm vững, tay tôi đã thử mua vài cuốn sách hướng dẫn, tôi cũng xem qua đĩa hình nhưng tôi vẫn chưa có chút khái niệm gì về nó; Nhưng mọi người đều biết sử dụng, tôi lạc hậu rồi, phải làm sao bây giờ? Tôi thấy rất lo lắng".

NGUYÊN TẮC 17: XÂY DỰNG QUAN HỆ VỚI KHÁCH HÀNG, CHỨ KHÔNG ĐƠN THUẦN LÀ BÁN SẢN PHẨM

It's a relationship , not a sale

Lúc đó, do quan niệm về C++ còn mới mẻ, rõ ràng là không thể dùng ngôn ngữ lập trình phức tạp như vậy để phát triển thị trường. C++ thực sự rất khó, khách hàng không biết nên học sử dụng như thế nào, ngay cả chương trình Windows cũng vậy, rất nhiều người đều chán ngán. Họ thấy rằng muốn làm được phần mềm xuất sắc thì phải bỏ quá nhiều công sức.

Thế là trong nhóm có người nghĩ ra biện pháp thông minh: *Thuật sĩ* (wizard). Hiện nay thì nó không có gì xuất sắc, nó không phải gậy thần thông, chẳng qua chỉ là cái nút để tạo ra một đoạn mã mà thôi. Xét về khái niệm, thuật sĩ chẳng khác gì phần dễ nhất trong số các tính năng mà mình đã sử dụng (tuy cơ sở của nó là thư viện Microsoft-MFC, rất khó). Nhưng thuật sĩ chính là hồng tâm, nó giúp các lập trình viên chỉ cần bấm một nút là có được một đống mã cơ bản.

Sau khi chúng tôi đưa *thuật sĩ* vào sản phẩm, tỷ lệ chiếm lĩnh thị phần của Visual C++ tăng rõ rệt lên hàng trăm điểm. Từ ví dụ của Visual C++ có thể thấy,

ứng dụng kỹ thuật cao không phải là trọng điểm, mà trọng điểm chính là giúp khách hàng của bạn, tìm ra nhu cầu thực sự của họ, họ đang băn khoăn về những vấn đề gì, sau đó tập trung tất cả chú ý vào nó. Không cần quan tâm đến mẫu hay các danh từ kỹ thuật khác, chỉ cần toàn tâm toàn ý phát triển những thứ mà đa số mọi người đang cần là được.

Khách hàng cần được thấu hiểu, tuy có thể họ không thể nói ra, bạn cần phải cảm nhận nhu cầu thực sự, đồng thời dùng sản phẩm phần mềm để chứng minh là bạn đã hiểu khách hàng. Đặt trọng tâm kỹ thuật vào đây, làm ra những thứ mà khách hàng mong mỏi. Trong ví dụ Visual C++, mong muốn của khách hàng là: "Tôi muốn trở thành cao thủ trong lập trình C++, trở thành lập trình viên với mức lương 90 ngàn đô Mỹ/năm. Nếu anh làm cho tôi thực hiện được mong muốn này, mà không phải học mệt như vậy, tôi sẽ bỏ tiền mua sản phẩm của anh, đồng thời bảo các bạn tôi đều dùng sản phẩm này, tôi chắc chắn sẽ mua nó! Nếu công ty không mua tôi sẽ tự móc túi ra mua, nếu vợ tôi không đồng ý tôi sẽ tranh luận đến cùng, tôi sẽ mua nó với bất cứ giá nào".

"Tôi chắc chắn sẽ mua phần mềm này! Dù có phải cãi nhau với vợ, tôi nguyện mua nó với bất cứ giá nào".

Là hãng sản xuất phần mềm, cần luôn ghi nhớ khách hàng cần cảm giác an toàn, mong có thể điều khiển được phần mềm và mong nó có thể giúp họ thành công. Khách hàng không thừa nhận họ không thể theo kịp khoa học kỹ thuật mới, họ sẽ không nói: "Phần mềm của anh hỏng rồi". Họ sẽ nghĩ: "Chắc là mình dốt quá". Thông thường người ta tin rằng mình đủ thông minh để nắm vững kỹ thuật, còn không đều cảm thấy sợ hãi hoặc né tránh đối với khoa học kỹ thuật. Bạn không thể chờ đối thủ làm cho khách hàng thông minh.

Quan hệ của bạn với khách hàng cũng như bạn nhảy. Bạn tiến lên trước một bước (Đưa ra sản phẩm mới và truyền đạt thông tin), sau đó khách hàng hưởng ứng một bước, bạn lại đưa một bước; Bạn phải chú ý đến chỉnh thể chứ không chỉ là bước vừa đi. Đó chính là kế hoạch nhiều phiên bản mà tôi đã đề cập trong nguyên tắc 3. Nếu khách hàng biết bạn ra phiên bản mới đúng hạn, và bạn trung thành đi theo họ trên con đường khoa học kỹ thuật dài dằng dặc này, họ sẽ phối hợp với bạn ở bước tiếp theo.

Quan hệ của bạn với khách hàng cũng như tình nhân. Nếu bạn bỏ quên quá lâu, hoặc biểu hiện không có hứng thú hay cự tuyệt, đương nhiên quan hệ của bạn sẽ xấu đi. Bạn bỏ quên sẽ làm khách hàng cảm thấy bị

lừa dối và tổn thương, thậm chí bực mình, tiếc là không "báo ứng" cho bạn thôi. Do đó phải trung thành tuyệt đối với khách hàng.

CHẮC LÀ TÔI DỐT QUÁ

Cách đây không lâu tôi có nói chuyện với một chị trên máy bay, chị ta hỏi tôi làm gì. Thông thường khi mọi người nghe nói tôi làm ở Microsoft, thường có hai kiểu phản ứng, một là tỏ vẻ rất ngưỡng mộ, hai là hỏi thẳng tôi có biết Bill Gates không.

Sau khi chị hỏi về Bill Gates, bắt đầu nói chuyện với tôi về máy tính, nói cho tôi biết kế hoạch học vi tính của chị. Chị ta nói rằng muốn đến học tại một trường đại học về tất cả các phần mềm máy tính, bao gồm dữ liệu, bảng tính và xử lý văn bản. Nói đến đây mặt chị lộ rõ vẻ phấn khích: "Máy tính thật tuyệt, tôi muốn biết mọi thứ: Windown, RAM... dù sao cũng phải học hết, dù mất vài năm cũng không sao"

Nghe thấy giấc mơ của chị, tôi có hai suy nghĩ về thái độ của chị đối với cách mạng máy tính. Tôi khâm phục dũng cảm, quyết tâm và sự nhìn xa của chị, không chịu lùi bước trước khoa học kỹ thuật. Chị chấp nhận đầu tư thời gian, tiền bạc và sự nỗ lực để học hỏi về vi tính, không sợ chấp nhận sự thực là mình không biết máy tính.

Ồ! Đợi đã, chị ta thực sự thấy máy tính rất khó, nhưng có thật sự khó như vậy không? Chị phải quay lại trường mới học được sử dụng các phần mềm trên PC. Nhưng thực tế các phần mềm này rất dễ. Nói như vậy thì chị ta chắc rất dốt, quay lại làm sinh viên lần nữa chỉ để học vi tính. Đúng rồi, các hãng phần mềm đã vớ bẫm, làm phần mềm thật khó để tỏ ra trí tuệ của mình hơn người khác một bậc, sau đó khách hàng phải bỏ tiền ra để học phần mềm. Dù phần mềm của chúng tôi rất dở, khách hàng cũng chỉ cảm thấy mình thật dốt hoặc vi tính khó quá.

Bạn đã là độc giả của cuốn sách này, thì cũng có kiến thức chuyên ngành nhất định, hơn nữa bạn bè bạn cũng biết rằng bạn có hiểu biết về vi tính. Hãy nghĩ xem, bạn có thấy nhiều người bảo rằng bạn chẳng biết gì về máy tính không? Đó cũng là hiện tượng tương tự. Đa số mọi người cho rằng mình không hiểu máy tính là do mình quá dốt, và cảm thấy lo lắng. Mọi người có cảm giác sai lầm là: Máy tính đúng, còn mình thì sai.

NGUYÊN TẮC 18: TĂNG CHU KỲ SẢN PHẨM

Cycle rapidly

Chúng ta đã nói về việc đưa sản phẩm phần mềm ra đúng hạn, bao gồm cả tiến độ và chất lượng. Ngoài ra, bạn còn phải tăng tốc độ đưa ra phiên bản mới, bởi các lý do sau: Thứ nhất, dù bạn có làm nhanh đến đâu, thị trường, sự tiến bộ của khoa học kỹ thuật và đối thủ đều có thể nhanh hơn; Thứ hai, quan hệ giữa bạn và khách hàng phụ thuộc vào việc bạn có thường xuyên liên hệ với họ không, vật dùng để liên hệ chủ yếu là phiên bản mới. Bạn cần cho khách hàng thấy vấn đề gì cũng có trong phần mềm: Quan tâm của bạn tới khách hàng, nhiệt tình của bạn, tâm niệm của bạn, cách nhìn nhận của bạn với khách hàng, đều biểu hiện trên phần mềm, nó sẽ không bao giờ nói sai, nó chính là đại diện của bạn.

Tôi chưa hề nghe thấy bất cứ quan hệ nào do thành tâm liên hệ mà bị tan vỡ. Liên tục đưa ra phiên bản mới, không chỉ làm tăng uy tín của bạn trên thị trường, cũng như thông tin phản hồi của khách hàng rút ngắn khoảng cách với khách hàng. Thường xuyên đưa ra phiên bản mới còn làm tăng lượng giao dịch, làm cho quan hệ hai bên trở nên chặt chẽ.

Nhóm Visual C++ có nhiều kinh nghiệm phong phú về việc đưa ra phiên bản mới trong thời gian ngắn. do đó chúng tôi rất thành công trong phương diện này. sau này vẫn còn tiếp tục. Đó là vũ khí cạnh tranh rất lớn của chúng tôi. đồng thời biểu thị chúng tôi có khả năng duy trì quan hệ tốt đẹp với thị trường. Hiện nay chúng tôi còn có thể dùng phương pháp dự tính để bán phần mềm, mỗi năm đưa ra 3 phiên bản mới. Tôi nghĩ rằng đây là phương thức bán phần mềm tốt nhất. nhưng có hai vấn đề: Thứ nhất, rất ít hãng có thể đảm bảo đưa phiên bản mới ra nhanh chóng và theo chu kỳ; Hai là nâng cấp phần mềm sẽ phải bỏ một lượng vốn khá lớn. và việc quản lý cũng rất khó khăn. Nhưng tôi tin rằng các hãng phần mềm hiện nay càng ngày càng có khả năng giải quyết vấn đề này. hơn nữa sự chín muồi của PC và mạng Internet sẽ dần phổ cập phương thức này.

Đứng trên lập trường của khách hàng, họ không nhất thiết phải sử dụng tất cả các phiên bản, có thể họ đã quen với môi trường cũ. hoặc do hạn chế tài chính. hoặc đã đầu tư nhiều cho phiên bản cũ (Chẳng hạn phát triển một số phần mềm ứng dụng). Dù lý do nào, khách hàng luôn có quyền nói cần hay không cần. Khách hàng có kế hoạch về kỹ thuật, chi tiêu hay sử dụng thời gian của riêng mình, họ cũng cần biết lúc nào bạn đưa ra phiên bản mới, để họ có thể sử dụng những

kỹ thuật mới nhất trên thị trường. Đáng tiếc là, các phần mềm hiện nay đều được nâng cấp không định kỳ, mà phiên bản mới thường có thay đổi lớn hoặc hàng loạt lỗi.

Thường xuyên đưa ra phiên bản mới đúng hạn (Quy mô sửa chữa nhỏ) còn có hiệu quả hơn việc đưa ra phiên bản thay đổi lớn không định kỳ, hơn nữa lợi nhuận nhiều hơn. Điều quan trọng nhất trong phát triển phần mềm là có phản ứng nhanh đối với thị trường, chứ không phải đột phá có tính cách mạng. Định kỳ đưa ra bản nâng cấp giúp bạn có cơ hội tác động trực tiếp đến nhu cầu người tiêu dùng, đồng thời nhanh chóng loại bỏ các chướng ngại khó chịu, và khách hàng sẽ thầm cảm ơn bạn. Loại bỏ ưu phiền cho khách quả thực rất có lãi, thường không cần nhiều kỹ thuật khó, mà đôi khi chỉ là thói quen sử dụng. Như vậy, khách hàng sẽ tin rằng bạn có thể giải quyết vấn đề cho họ, tặng thêm cảm giác an toàn.

Tôi thấy rằng khách hàng rất thích sản phẩm mới bày ra trước mắt họ, chứ không cần sự cam đoan của hãng, chỉ có rất ít người cần cả hai điều trên. Nhưng thông thường, **liên tục đưa ra phiên bản mới là bằng chứng tốt nhất đối với khách hàng**. Tặng quà chỉ làm cho họ vui mừng trong giây lát, chưa chắc làm họ thích mua sản phẩm của bạn.

Hoàn thành nhiều thành công nhỏ còn được mọi người chấp nhận hơn là chỉ có một thành công lớn. Vì phương thức này khá thích hợp với nhu cầu của người sử dụng. Đưa ra phiên bản mới đúng hạn sẽ giúp bạn đưa ra mục tiêu nhỏ, dễ hiểu và dễ nắm bắt hơn, đồi thời dễ quản lý hơn mục tiêu của phiên bản lớn.

THIẾT KẾ

Đối với một phần mềm lớn, điều quan trọng nhất là đưa ra sản phẩm đúng thời cơ. Cũng chính là, bạn phải biết cách đưa sản phẩm ra đúng hạn, đồng thời nắm được nhu cầu sâu thẳm trong lòng khách hàng, như vậy càng hiểu được nội tâm của khách, phần mềm sẽ càng lớn. Thiết kế phần mềm - mỗi thành viên đều phải tham gia - biểu thị mức độ hiểu về nhu cầu tính năng của cả nhóm. Tóm lại, bí quyết đầu tiên trong thiết kế phần mềm là: Kết hợp các biện pháp hay nhất của nhóm lại, nhằm đáp ứng nhu cầu sâu thẳm của khách hàng.

Trong quá trình thiết kế, khó khăn nhất là để cho biện pháp, kiến nghị hay nhất của một người được biểu hiện ra. Nhưng làm như vậy cũng rất đáng giá. Thử nghĩ xem: Nếu chỉ có 2 hay 3 người thiết kế phần mềm, IQ tổng cộng chỉ có 230 hay 250, nếu có 10 người tổng hợp có thể lên tới 1300. Đếm xem nhóm của bạn có bao nhiêu người, IQ kết hợp càng cao khả năng làm ra phần mềm lớn càng cao. Trong mỗi giai đoạn thiết kế hay phát triển, để phát huy sáng tạo là điều rất quan trọng. Một toà nhà lớn chỉ cần một kiến trúc sư giỏi, nhưng thiết kế phần mềm phải cần đến hàng trăm hàng ngàn người sáng tạo giá trị cho nó.

Nhưng, để mọi người được phát huy sức sáng tạo mà không ảnh hưởng đến các nhân tài khác là một việc rất khó, cũng là thách thức lớn nhất của lãnh đạo, tuy đây không phải là thách thức duy nhất trong ngành phần mềm. Giáo dục là vai trò chính của lãnh đạo hay quản lý: Dẫn dắt nhóm thảo luận nghiên cứu, dẫn dắt mọi người suy nghĩ cách giải quyết mọi vấn đề, làm cho mọi việc đều được làm tốt và đúng lúc đúng chỗ.

NGUYÊN TẮC 19: TÌM KIẾM SỰ VƯỢT TRỘI

Go for greatness

Làm thế nào để thành viên sống trong môi trường tốt đẹp, bồi dưỡng tố chất của họ với vẻ đẹp? Cho thành viên đọc sách báo gì hoặc thách thức như thế nào, để có thể tăng sự trưởng thành cho họ? Quản lý cần bồi dưỡng khái niệm Mỹ học cho thành viên, có thể lấy văn minh nhân loại mấy ngàn năm làm nguồn cho Mỹ học, với trọng điểm là thiết kế phần mềm. Cảm thụ của mọi người với cái đẹp rất phức tạp, vài thế kỷ nay luôn có người nghiên cứu chủ đề này. Do đó, tốt nhất sử dụng kinh nghiệm của người đi trước làm nền tảng, dưới đây là một số lý luận dùng để thảo luận trong chương này.

Ngoài Mỹ học, chúng ta có thể sử dụng vai trò của lịch sử: Tạm thời quên đi thời đại thông tin này, xem trong lịch sử có nhân vật hay sự kiện nào làm bạn cảm động hay ngưỡng mộ, thay đổi cách nghĩ hay thái độ của bạn? Công việc của nhóm bạn là gì? Hai thứ có gì khác biệt? Câu chuyện lịch sử này có kích thích tiềm năng của mọi người, đồng thời mở ra một cánh cửa sáng tạo mới? Sáng tạo này có phải chưa có ai nghĩ ra? Đồng thời dẫn đến một tương lai chưa ai nghĩ đến? Bạn có thể suy nghĩ các vấn đề này, dùng con mắt lịch sử phân tích công việc của bạn, có thể sẽ có kết quả ngoài mong đợi.

NGUYÊN TẮC 20: THIẾT LẬP CHỦ ĐỀ

State your theme

Có một số tác phẩm lớn đã ảnh hưởng tới quan niệm của tôi về các phần mềm lớn, đó là: "*Mỹ cảm*" (The Sense of Beauty) của George Santayana xuất bản năm 1896, và "*Quy tắc Mỹ học*" (Algorithmic Aesthetics) của George Stiny và James Gip xuất bản năm 1978. Nhất là tác phẩm thứ hai, có liên quan lớn đến thảo luận dưới đây của chúng ta đối với Mỹ học phần mềm. Định nghĩa của George Stiny về Mỹ học như sau:

> *Các vấn đề Mỹ học quan tâm là miêu tả, giải thích và đánh giá một tác phẩm nghệ thuật như thế nào, và làm thế nào để sáng tạo một tác phẩm nghệ thuật đẹp.*

Trong tác phẩm "*Phân tích nghệ thuật*" (The Analysis of Art) của Dewitt H. Parker năm 1926 có đề cập đến 6 chuẩn tắc của Mỹ học, rất thích hợp để phân tích thiết kế phần mềm, tôi đã yêu cầu nhóm của chúng tôi lấy làm tiêu chuẩn thiết kế.

> *Sáu chuẩn tắc Mỹ học của Parker là: Thống nhất, chủ đề, biến tấu, diễn biến, cân bằng và tầng lớp.*

Theo quan điểm này, *tính thống nhất* (Unity) là nguyên tắc quan trọng nhất của tác phẩm nghệ thuật lớn. Tôi đã thấy thuộc tính này trong rất nhiều phần mềm lớn. Nếu chúng ta lấy cách giải thích tính thống nhất của Parker áp dụng vào phần mềm, có thể nói, một phần mềm có tính thống nhất, giá trị Mỹ cảm của mỗi bộ phận đều rất quan trọng đối với chỉnh thể, và mỗi nguyên tố Mỹ cảm đều cần phải xuất hiện trong tác phẩm, không thể thiếu bất cứ cái nào, và những thứ không nên xuất hiện hay không có cống hiến đối với Mỹ cảm thì không được xuất hiện. Do đó, thiết kế phần mềm cần phải đưa tất cả những thứ khách hàng cần vào, và bỏ đi những thứ họ không cần, do bất cứ thứ gì trong phần mềm đều cần đến, nên khi sử dụng không được để khách hàng bị quấy nhiễu, làm phải để ý đến những thứ không cần thiết. Do đó, mong muốn tính thống nhất của phần mềm, là mục tiêu quan trọng trong thiết kế phần mềm. Có thể trong lĩnh vực khác (Chẳng hạn thiết kế một tác phẩm nghệ thuật) đã từng dùng trực giác hay lý luận khác để vận dụng tính thống nhất, còn tôi thì thấy sáu nguyên tắc của Parker rất hữu dụng trong thiết kế phần mềm.

Chủ đề của phần mềm (Theme) sẽ chi phối quan điểm cơ bản trong thiết kế, và cũng là nguồn gốc chính của giá trị phần mềm. Do đó, bạn phải truyền tải rõ ràng chủ đề này trong nhóm, để nhân viên phát triển

và nhân viên kinh doanh hiểu thật rõ chủ đề mới được. Về thực chất chủ đề phần mềm là từ đồng nghĩa với mục tiêu. Mục tiêu càng rõ ràng, lực đẩy gây ra càng lớn, vì bạn có thể giảm thấp sự mơ hồ, mà cảm nhận về mục tiêu của mọi người càng thống nhất hơn, cả quá trình thiết kế càng thuận lợi. Nhưng sau khi quyết định chủ đề, bạn còn phải chú ý cắt bỏ tất cả các phần không liên quan đến chủ đề, dù nhân viên phát triển cho rằng phần đó rất quan trọng, hoặc đó là niềm tin của bạn, đều cần phải cắn răng chịu đau vứt đi.

Thông tin tiêu thụ sản phẩm có thể sinh ra từ chủ đề. Các nhà quan sát giỏi chỉ cần xem chủ đề là nắm được thông tin. Thông tin chỉ là bổ sung nói rõ ý nghĩa chủ đề, nếu chủ đề mơ hồ hoặc không chỉ là một, thông tin kinh doanh có tốt đến đâu cũng vô dụng (Mọi người đều biết rõ đạo lý này! Sản phẩm mà không có hình tượng duy nhất và rõ ràng, quảng cáo nhiều đến đâu cũng không có tác dụng). Chủ đề sản phẩm bắt nguồn từ quan niệm của bạn đối với thị trường, từ ví dụ Visual C^{++} của chúng tôi có thể thấy quan niệm đối với thị trường: Ai cũng thấy Visual C^{++} khó học, do đó chúng tôi thiết kế chủ đề: "Để người dùng sử dụng Visual C^{++} dễ dàng", và thông tin tự nhiên của chúng tôi là: Visual C^{++} làm cho C^{++} dễ dàng hơn.

Trọng điểm là tính năng sản phẩm không phải những thứ trong túi tóm lấy lúc nào cũng được, cần loại bỏ những thứ không liên quan với chủ đề, và mục tiêu phải phù hợp tính thống nhất mới được, điểm này cộng với chủ đề trở thành hai mặt của một thể thống nhất. Đầu tư vốn vào mục tiêu này, làm cho tất cả mọi người đều hiểu rõ mục tiêu, đồng thời nỗ lực vì mục tiêu. Làm được như vậy, sản phẩm của bạn sẽ bao hàm mục tiêu này.

CHUYÊN TÂM VÀO CHỦ ĐỀ

Tôi không biết câu chuyện dưới đây có thật không, nhưng tôi rất thích và thường kể cho các thành viên nghe:

Một công ty chuyên về bảng tính điện tử đã nghiên cứu thấy hầu hết người dùng sau khi gõ 20 số là muốn dùng các con số đó để vẽ biểu đồ. Nghiên cứu cho thấy, đa số mọi người khi dùng bản tính điện tử đều có hành vi như vậy.

Do đó họ nghiên cứu nhiều lần, phát triển đi phát triển lại, toàn tâm toàn ý để sản phẩm có chức năng này: Chỉ cần sử dụng rất ít con số là có thể dễ dàng vẽ được biểu đồ. Những người dùng vốn sử dụng bảng tính điện tử vẽ biểu đồ thấy rất hứng thú với tính năng này, cho rằng có chức năng này thì cũng đáng mua sản phẩm của họ.

Cũng có một câu chuyện khác gần như vậy, liên quan đến nhóm xây dựng phần mềm tài chính gia đình. Sau khi nghiên cứu thị trường họ phát hiện: Sản phẩm này phải đem ngay lại lợi ích cho khách hàng, nếu không sẽ không bán được, và họ cũng sẽ không mua bản nâng cấp. Họ quyết định làm cho bất cứ người dùng nào kể cả không biết vi tính, chỉ sau 10 phút mở hộp thấy ngay hiệu quả, từ lắp đặt, thao tác, nhập xuất đều phải vừa đơn giản vừa nhanh.

Do đó họ cử người đến cửa hàng phần mềm, sau khi được khách hàng đồng ý, theo họ về nhà và quan sát tình hình sử dụng của họ, ghi chép lại cẩn thận: Kể cả sau khi mở hộp ra trước hết họ xem có cái gì, trong quá trình cài đặt và chạy phần mềm gặp phải những khó khăn gì,... để làm căn cứ cải tiến sản phẩm. Sau khi phân tích nghiên cứu, họ tìm được hàng chục cơ hội tăng sự hài lòng của khách hàng, và họ dần thực hiện mục tiêu này trong vài phiên bản tiếp theo - Thoả mãn trong 10 phút. Cuối cùng sản phẩm đã thành công!

Biến tấu (Variation) là sau khi hơi thay đổi tô điểm cho chủ đề, biểu hiện lại lần nữa. Sau khi đã biểu hiện chủ đề, để tiếp tục thu hút sự chú ý và hứng thú của người dùng, dùng phương thức khác - biến tấu - để giải thích chủ đề, tăng cường nhận thức của người dùng với chủ đề, lưu lại ấn tượng sâu sắc.

Diễn tiến (Evolution) có nghĩa là dùng phần trước để quyết định phần sau, cũng như quá trình học trước hết phải nhập môn sau đó mới đi sâu, từ nông đến sâu giúp người ta dễ dàng tiếp nhận, càng thích học hơn. Nếu phần mềm trước sau có thể như vậy sẽ có được kết quả tốt đẹp.

Cân bằng (Balance) là không quá coi nhẹ hay nhấn mạnh bất cứ bộ phận nào trong phần mềm. Chẳng hạn hai đối tượng tương phản nhau cần phải hướng dẫn đều nhau.

Tầng lớp (Hierachy) chỉ các nguyên tố trong phần mềm, tuỳ thuộc vào tính quan trọng và kích thước mà có tỷ trọng hợp lý. Quan niệm tầng lớp và cân bằng khá gần, có thể nói tầng lớp là phương pháp xây dựng với cân bằng. Nếu chủ đề ở đầu của tầng lớp, thì phải cân bằng giữa các bộ phận của tầng dưới, sự hỗ trợ chủ đề của các bộ phận cùng tầng lớp cũng phải ngang nhau, càng gần đỉnh sự hỗ trợ càng lớn.

THUỘC TÍNH CỦA CÁI ĐẸP

Năm 1975, trong tác phẩm "Học thuyết mới về cái đẹp" (A new theory of beauty) của Guy Sircello đã đưa ra học thuyết về cái đẹp khá thú vị. Chúng ta không bàn về cách phân chia chi tiết về chất và lượng của đặc tính vật thể mà là một số lý luận về cảm thụ cái đẹp. Ông cho rằng, sở dĩ con người cảm nhận được cái đẹp, vì vật

thể đó có một thuộc tính đẹp trở lên; Ông phân tích sâu hơn, chỉ có một thuộc tính được nhấn mạnh trong tác phẩm mới có thể có thuộc tính đẹp, mà một vật thể phải có từ một thuộc tính trở lên, mọi người mới cảm nhận được cái đẹp của nó. Thuộc tính đẹp không đảm bảo toàn bộ vật thể có vẻ đẹp, nhưng vật thể không có thuộc tính đẹp đẽ sẽ không thể đẹp được.

Học thuyết của Sircello ít nhiều giải thích tại sao một số phần mềm không được coi là tốt, các phần mềm thất bại như vậy ở đâu cũng thấy, có hàng loạt tính năng sản phẩm nhưng không có chủ đề, kết quả là không có tính năng nào làm phần mềm nổi trội, không gây được sự chú ý của khách hàng, đương nhiên như vậy là thất bại.

NGUYÊN TẮC 21: KHÔNG DỰA VÀO NHỮNG VIỆC KHÔNG XÁC ĐỊNH

Minimize dependencies

Cố gắng giảm thiểu những việc mà nhóm cần nhưng không thể khống chế. Khi bắt đầu dự án, những việc quyết định dựa vào càng ít, thì sau này càng thuận lợi. Thông thường, hiệu suất thiết kế chương trình không thể cao quá (Có lẽ do không thành thạo hay không bắt được lỗi), có thể bạn chịu khó làm thêm để hoàn thành công việc của mình đúng hạn, nhưng người khác chưa chắc làm được như vậy.

Do đó, khi thiết kế phải suy nghĩ những việc cần thiết nhưng chưa xác định, cần biết rằng chúng có thể ngốn rất nhiều vào giá thành, chỉ trong trường hợp rất quan trọng hoặc bất đắc dĩ mới cho phép sự dựa dẫm này, để các thành viên hiểu rõ hậu quả nghiêm trọng của nó, cố gắng phối hợp với nhau, và đánh giá khả năng mất kiểm soát của các việc này, và có thể có ảnh hưởng gì.

NGUYÊN TẮC 22: GIẢM SỰ PHẪN NỘ CỦA KHÁCH HÀNG

Propitiate the goods

Trong quá trình tiến triển của dự án, thường vẫn có một số nhân tố có tính dựa dẫm hay có tính bất định bám lấy bạn. Bạn phải tìm ra nhân tố làm vướng chân bạn nhất, nghiên cứu xem nếu có vấn đề phát sinh thì phải làm những gì.

Hãy kiểm tra các tính năng sản phẩm, xem có phần nào không rõ ràng hoặc không có ý nghĩa lớn trong việc thoả mãn khách hàng, khi không còn nhiều thời gian thì hãy cắt bỏ. Đúng vậy, có thể có khách hàng không vừa lòng vì bạn không làm tính năng này, nhưng chỉ cần bạn đưa phiên bản mới ra đúng hạn, khách hàng đó sẽ không bỏ rơi bạn, đồng thời chờ đợi tính năng mà anh ta cần trong phiên bản tiếp theo.

NGUYÊN TẮC 23: KHẢ NĂNG CẤY GHÉP CỦA PHẦN MỀM

Portabilily is for canoes

Đối với đa số các hãng phần mềm, hỗ trợ *đa nền* (platform) rất khó khăn. Dù có không quan tâm đến việc tăng giá thành do hỗ trợ tăng thêm, công việc về bảo đảm chất lượng cũng tăng rõ rệt, dù quản lý bảo đảm chất lượng có xuất sắc nữa thì cũng khó có thể giải quyết vấn đề này. Cách tốt nhất là yêu cầu hãng phần mềm hệ thống cung cấp công cụ hỗ trợ, sau đó cẩn thận quyết định nền mà bạn hỗ trợ, hạng mục càng ít càng tốt.

Nhưng chớ có chọn nhầm, nó có thể dẫn bạn tới thất bại.

NGUYÊN TẮC 24: KHI THIẾT KẾ CẦN SUY NGHĨ ĐẾN NHÂN TỐ THỜI GIAN

Design time at design time

Khi thiết kế phải suy nghĩ đến nhân tố thời gian, không được thiết kế xong mới quyết định xem phải mất bao lâu mới làm xong. Thời gian là điều kiện hạn chế lớn nhất của bạn.

Phải hoàn thành mục tiêu thiết kế sản phẩm mới được đưa ra, bạn không thể hoàn thành một nửa rồi trao cho khách hàng. Nhân viên phát triển và quản lý khi thiết kế sản phẩm rất dễ bỏ quên nhân tố thời gian. Cách làm đúng đắn hoàn toàn ngược lại, trong giai đoạn thiết kế phải coi thời gian là nhân tố mấu chốt, khi suy nghĩ phương án thay thế, thời gian ít thì thêm, thời gian nhiều thì giảm. Thông thường chỉ cần coi nhân tố thời gian là trọng tâm suy nghĩ khi thiết kế, là bạn có thể rút ngắn thời gian phát triển.

Nếu thiết kế của bạn không cần phải hoàn thành đúng hạn? Đừng có như vậy, hoàn thành đúng hạn là quan trọng nhất, còn quan trọng hơn bất cứ một lý tượng lớn (Đáng tiếc không thực hiện được) nào khác.

PHÁT TRIỂN

Phát triển bàn đến ở đây, bao gồm cả các hành động thực tế sau: Kế hoạch thực hiện, sắp xếp tiến độ, viết chương trình và kiểm tra chất lượng; Chứ không phải bàn về việc ai làm gì, trong nhóm ai cũng phải tham gia hoạt động phát triển, do đó ai cũng là nhân viên phát triển.

Mọi người dùng phát triển phần mềm để biểu thị quá trình sản xuất phần mềm. Điều này rất thú vị, nó biểu thị hình thành của phần mềm là một quá trình trưởng thành dần dần, qua các bước có thứ tự và ảnh hưởng lẫn nhau. Rốt cuộc nhân tố nào thúc đẩy sự trưởng thành của phần mềm? Là tập trung sáng tạo của nhóm. Từ những tài sản trí tuệ của nhiều người, dần dần kết hợp tạo thành bộ tài sản chặt chẽ mà hoàn chỉnh, đó chính là sản phẩm phần mềm sắp ra đời. Điều này lại làm tôi nhớ đến chủ đề của cuốn sách này: Hoạt động cơ bản của phát triển phần mềm là kết hợp trí tuệ của nhiều người thành một khối tài sản trí tuệ.

Về thực chất điểm kết thúc của phát triển phần mềm là cực hạn trí tuệ của nhân viên phát triển.

Nhưng quá trình phát triển phần mềm liên quan đến rất nhiều khía cạnh. Mỗi cá nhân, đoàn thể và khoa học kỹ thuật, cả ba đều không ngừng phát triển, không phải là cố định bất biến. Do đó, phát triển phần mềm không có điểm khởi đầu và điểm kết thúc rõ rệt, mà là rất nhiều người hợp vào theo các tầng cấp khác nhau, cùng nhau sáng tạo. Tuy mỗi phiên bản mới, đều có thể nói là một cột mốc nhưng không thể coi nó như điểm kết thúc, thực tế điểm kết thúc trong phát triển phần mềm là (Cực hạn trí tuệ) nhân viên phát triển.

Rất nhiều chuyên gia dùng con mắt của công trình để phân tích vấn đề phát triển phần mềm, mà vấn đề phát triển phần mềm thường được coi là vấn đề công trình. Nhưng tôi lại thường coi phát triển phần mềm như hiện tượng văn hoá hay xã hội học. Một nhóm người cùng sáng tạo tài sản trí tuệ, còn khó hơn phương pháp luận thiết kế công trình hay các lý luận cơ bản khác, tình hình của công trình là hữu hạn, nhưng các vấn đề trong quá trình phát triển phần mềm là vô hạn, hơn nữa không có sự đúng sai rõ ràng, rất nhiều vấn đề được giải quyết dựa vào sự hoà hợp và chọn hay bỏ. Thông minh tài trí là điều kiện cần tuyệt đối để sản xuất phần mềm, nhưng chỉ có thông minh tài trí vẫn chưa đủ.

Phát triển trong thời kỳ bố cục, cũng như phải nhóm phương trình lập phương vậy, hàm chứa thứ tự thời gian không thể dự đoán và nội dung sản phẩm không xác định. Cần biết rằng, trước khi nói hết mà vẫn chưa thực sự bắt tay làm, không ai có thể biết hình dạng phần mềm khi hoàn thành. Trong giai đoạn này, tính năng sản phẩm chỉ là tên gọi mô tả, gần như chức năng kỳ vọng, còn thiết kế thường chỉ thực sự kết thúc khi phiên bản mới đã ra lò. Nói cách khác, đại lượng biến đổi trong phương trình này là giá trị không xác định, đồng thời thay đổi theo diễn biến của giai đoạn phát triển phần mềm, đây là vấn đề rất phức tạp.

Nếu nói phát triển phần mềm là nhóm chương trình có kết cấu đặc biệt, không bằng nói đó là kết hợp mật thiết tri tuệ con người.

Còn một phương trình khó giải nữa trong giai đoạn bố cục là sáng tạo của con người, cũng chính là lúc bạn phải đánh giá mức độ sáng tạo của mỗi thành viên. Chúng ta không chỉ cần suy nghĩ kỹ nội hàm của tính năng trong giai đoạn này, mà còn dự đoán xem nó chuyển thành một dự án cụ thể rõ ràng như thế nào, lúc nào có thể đưa vào tay thành viên để gia công và hoàn thành.

Nhân tố có tính bất định nêu trên thực sự rất nhiều và rất phức tạp, làm cho sắp xếp tiến độ hầu như là nhiệm vụ không thể thực thi. Đương nhiên, bạn còn phải thiết lập mục tiêu chỉnh thể cho dự án, sau đó tập trung tinh thần vào cột mốc đầu tiên, có thể vào một hai tháng đầu sau khi bắt đầu dự án, đây là việc mà bạn có thể làm được nhiều nhất trong giai đoạn bố cục. Trong phần 2 giai đoạn giữa, chúng tôi sẽ nói chi tiết kỹ thuật thiết lập cột mốc, đây chính là cơ hội tốt để nhóm luyện tập làm được thành quả cụ thể trong một thời gian ngắn. Chú ý khoảng cách mỗi cột mốc không được vượt quá 3 tháng.

Thông thường, phát triển phần mềm cũng như đuổi kịp tiến độ chứ không giống nhạc giao hưởng; Nhạc giao hưởng nhịp nhàng và nho nhã, còn phát triển phần mềm như một đống công việc dồn dập đổ tới. So sánh như vậy thật đáng sợ, hãy coi nó như diễn tấu ngẫu hứng vậy! Bạn cần phải biết nên biểu hiện vào lúc nào, lúc nào nghỉ một lát, mọi người đều ở trong trạng thái động, hợp tác chặt chẽ với rất nhiều người, bất cứ hai âm điệu nào đều không mâu thuẫn nhau, làm chỉnh thể biểu hiện thành một quy luật đẹp đẽ. Trong mọi tình huống không xác định, đưa cả hoạt động lên cao trào, làm cho trình độ không thấp hơn nhạc giao hưởng, hơn nữa, trong thực tế bản thân nó là một loại niềm vui.

NGUYÊN TẮC 25: TỪ CHỐI MỆNH LỆNH KHÔNG HỢP LÝ

Don't accept dictation

Tôi ngạc nhiên là có những nhóm phát triển phần mềm chấp nhận sự chỉ huy của người không có chuyên môn, nhất là những vấn đề liên quan đến tiến bộ. Trong thế giới hiện đại với lĩnh vực khoa học kỹ thuật lại sử dụng phương pháp điển hình của xã hội cũ. Có một số công ty cho người không hiểu về phát triển phần mềm quản lý những việc quan trọng nhất - thời gian, tính năng và tài nguyên là ba tam giác vàng của phát triển phần mềm - thực là quá điên rồ. Đáng buồn hơn là, nhiều nhân viên chuyên nghiệp lại chấp nhận kiểu lãnh đạo thiếu tri thức và tính lôgíc này, coi đó như một thứ tự tiêu chuẩn.

Tôi đã tiếp xúc rất nhiều nhóm phát triển, theo đánh giá riêng của tôi, có khoảng 30-40% số nhóm phải chịu những chỉ huy bất hợp lý như vậy. Cách làm hợp lý là người làm việc ước lượng thời gian. Đương nhiên, nếu chính xác không phải là một trong những mục tiêu thì ai ước lượng thời gian cũng được.

Nếu cấp trên không cho những người thực sự phụ trách nhiệm vụ ước tính thời gian, thì đó là sự chuyên chế.

Đôi khi, không ước tính được thời gian là vấn đề của nhóm, do nhân viên phát triển và phụ trách dự án không nhận trách nhiệm quyết định thời gian cần thiết đạt được mục tiêu. Tước quyền ước tính thời gian trên tay người thực thi công việc là cách làm không trao quyền. Trong bất cứ trường hợp nào, chấp nhận cách làm này đều rất đáng buồn, tiến độ mới bắt đầu đã là giả, nhóm làm sao có được năng lực làm việc nữa?

Không sai, chúng ta có thể đồng tình với một số cấp trên muốn cảm thấy mình có thể khống chế, có thể dự đoán, nhưng nguyên nhân nào làm những quyết sách mơ hồ đó được thực hiện, làm cho phát triển phần mềm trở thành một tai nạn lớn?

Bất kỳ là cá nhân hay tổ chức, đa số đều không thể rút kinh nghiệm từ trong sai lầm. Người ta có lúc sơ ý, lần trước làm sai, lần này thử lại xem có tốt hơn không, không hề suy nghĩ kỹ về nguyên nhân làm sai lần trước, thậm chí không thèm để ý xem những ví dụ thực tế về thành công trong phát triển phần mềm gợi mở cho ta những gì.

Hiện tượng mù quáng này dễ xảy ra trong nhóm có tiến độ bị tụt hậu nghiêm trọng, nhiều nhân viên phát triển không thể chịu nổi cảnh giẫm chân tại chỗ này nữa, chỉ có cách ra đi. Những người bỏ đi này có yêu cầu cao với công việc, mới không chấp nhận sống chung với sếp quan liêu như vậy, và thường họ cũng là những người quan trọng nhất. Những người ở lại thì ca thán, bị vứt vào đất hiểm mà không có ai quan tâm. Nhân viên kinh doanh như những thằng ngốc, lời hứa của họ như không khí, có thái độ buồn chán; Quản lý cấp cao thì đầy cáu giận, bực tức; Khách hàng cảm thấy mình bị lừa. Tai nạn thế là bắt đầu...

Cần biết rằng, trong viện tâm thần, người có đầu óc tỉnh táo lại bị coi là tâm thần.

Lâu dần, mây tan dần, mọi người lại nhen nhóm lên một tia hy vọng, có thêm thành viên mới, quản lý mới (Hoặc người bị kiểm điểm), kỹ thuật mới cũng thu hút nhiệt tình của nhân viên phát triển... Đã bị hố nhưng vẫn không học được gì, cấp trên lại quyết định ngày này tháng này phải làm xong phần mềm, vòng tuần hoàn tai nạn mới lại bắt đầu...

Nếu là bạn, phải làm gì để đối phó với tình hình sai lầm này? Nếu bạn phát hiện mình đang ở trong một

tổ chức như vậy, cần làm gì để đối mặt với khó khăn? Hãy ghi nhớ, trong môi trường sai lầm này, bất kỳ người bình thường nào đều có thể mất lý trí, mà trong tổ chức đang đầy rẫy những hành vi vô lý, bạn cũng không còn hy vọng gì khi ở lại, là người duy nhất tỉnh táo, chắc chắn sẽ bị mọi người cho là tâm thần. Tuy bạn có thể miễn cưỡng làm việc và bảo vệ mình trong môi trường đang tự huỷ diệt đó, nhưng chưa chắc bạn bạn đã làm như vậy.

Nhưng dù sao vẫn phải giải quyết vấn đề này. Do đó, khi nhận những mệnh lệnh vô lý, hãy tìm cách nhắc nhở "hôn quân" của bạn, làm như vậy là không đúng, không làm được như vậy do yêu cầu của hiện thực. Quyền lực không phải là vạn năng, cách làm của sếp không thông, bạn phải giúp anh ta hiểu ra, mọi người đang cố gắng hoàn thành mục tiêu của sếp, chứ không phải mục tiêu chung. Nếu như có quyền tham dự, tình hình sẽ tốt hơn, mọi người cũng vì thế mà làm việc nhanh hơn, vui vẻ hơn.

Bảng tiến độ công việc nên được làm từ cấp dưới rồi chuyển cho cấp trên, ai phụ trách công việc của người đó, cũng phụ trách thiết lập bảng thời gian riêng, đảm bảo hoàn thành công việc đúng hạn. Trách nhiệm và trao quyền là hai mặt của một thể, có cả hai mới có thể đưa ra kế hoạch phát triển hợp lý.

NGUYÊN TẮC 26: COI CÔNG VIỆC NHƯ TRÒ CHƠI

Now go play

Nếu coi phát triển phần mềm như một công việc, thì quả thực là một nhiệm vụ buồn tẻ; Xét từ góc độ khác, tôi khuyên bạn hãy coi nó như một trò chơi.

Trong giai đoạn bố cục, các sự việc đều tiến hành theo từng bước chưa chắc đã đảm bảo thành công.

Nếu bạn dùng các nguyên tắc của tôi để xây dựng nhóm cho bạn, tuy vẫn là phần mềm đó, nhưng bạn sẽ thấy họ tự nhiên hơn, phản ứng nhanh nhạy, hài hước hơn, dễ dàng tạo ra các phần mềm xuất sắc. Coi cạnh tranh như trò chơi, bạn sẽ dễ thắng hơn. Máy tính thực chất là một công cụ đầy thú vị và thách thức, còn nhân viên phát triển thì sao? Đương nhiên là người rất nghịch rồi.

Trong một nhóm lành mạnh, dùng phương thức trò chơi để làm việc là điều rất tự nhiên. Quản lý không cần có hành động nào đặc biệt, chỉ trách không hạn chế sự phát huy thành viên, đồng thời chú ý xem mọi người

có vui vẻ làm việc và tự nhiên không là được. Khích lệ thành viên vui vẻ, không sợ họ phân tâm, đây chỉ là nguồn sáng tạo mà thôi.

Trò chơi phát triển phần mềm, sẽ tự sản sinh ra các quy tắc, người làm tốt sẽ được thưởng, làm không tốt sẽ bị phạt. Quản lý không phải lo lắng về chính nghĩa và trọng tài, thưởng phạt của chính trò chơi còn hiệu quả hơn thưởng phạt do bạn nghĩ ra.

Trò chơi này đem lại cho bạn lợi ích gì? Chơi nhiều, luyện tập nhiều, cảm nhận sự hứng thú và yêu thích nó. Như vậy, hứng thú từ chính sáng tạo phần mềm sẽ làm tiêu tan sự cứng nhắc, ngạo mạn, và đó là điều tốt nhất.

Trong giai đoạn bố cục, các sự việc được tiến hành theo từng bước chưa chắc đã đảm bảo sự thành công. Mục đích của phần này là giúp bạn chuẩn bị tốt đội ngũ trước khi phát huy tiềm năng lớn nhất của nhóm. Chương này không phải là công thức thành công, mà là điều kiện tiên quyết. Nếu mọi việc trong giai đoạn bố cục đều được tiến hành theo nguyên tắc của cuốn sách này, thì sau này vẫn tiến hành như vậy, giúp nhóm của bạn trưởng thành.

PART Two

PHẦN 2
GIAI ĐOẠN GIỮA

Cuốn sách giúp độc giả xây dựng quan niệm rõ ràng, chia quá trình phát triển phần mềm thành 4 giai đoạn, nhưng trong thực tế vận dụng rất khó có thể làm như vậy, các giai đoạn đều có phần lặp lại của nhau. Trong phần này, tôi không chia thành chương, đoạn vì nội dung phần này có thể dùng các nguyên tắc giải thích sẽ rõ ràng hơn. Do đó không nên cho rằng các nguyên tắc ở giai đoạn nào thì chỉ thích hợp với giai đoạn đó. Trong thực tế, các nguyên tắc tôi đưa ra đều là quan niệm, phạm vi thích hợp là cả quá trình phát triển phần mềm, nhóm phát triển tốt nên luôn chú ý xem từng nguyên tắc có được thực hiện không. Còn các nguyên tắc đề cập đến trong giai đoạn đầu mà bạn làm được, bạn sẽ thấy càng dễ dàng thực hiện trong giai đoạn giữa, vì bạn là luyện tập các phương pháp này.

Mục đích chính trong giai đoạn đầu là tập trung nhóm và xây dựng mục tiêu chung, còn giai đoạn giữa là kỳ vọng và bất định đối với mục tiêu, mọi sự việc đều hỗn loạn và đầy những lo lắng về khả năng thất bại. Lúc này, nghị lực là quan trọng nhất. Tạm không quan tâm xem nhóm có trưởng thành không, tạm bỏ qua những trải nghiệm của bạn, hay những vĩ đại khi sản phẩm ra lò, giai đoạn giữa đầy những hỗn loạn đáng sợ, có thể bị tấn công bất ngờ vào bất cứ lúc nào hoặc những thất bại không hề nghĩ tới, nhưng bạn phải kiên quyết đi tiếp.

Có lẽ bạn sẽ có tâm lý sau: Mong rằng có con đường để tôi tránh khỏi tất cả, dự án nào cũng có lúc như vậy. Sự việc nào cũng không thể xác định, và thế là tâm tư thành viên xuống cấp, tiêu cực, chống đối, mỉa mai đều xuất hiện, vài tháng tới không thấy ngày nào mọi người chăm chỉ làm việc, chẳng khác gì các thành viên đều bị hoại huyết. Tai nạn này hầu như không bao giờ dứt, thật đáng sợ!

Hít thở sâu, và bắt đầu vào giai đoạn giữa!

NGUYÊN TẮC 27: PHƯƠNG PHÁP BÁC SỸ

Be like the doctors

Khi bệnh nhân uống thuốc cũng không có hiệu quả, bác sỹ thường bảo lưu bệnh tình; Tránh không cho người bệnh quá bi quan hay sợ hãi, đồng thời khích lệ người bệnh giữ hy vọng, tốt nhất cho họ có một mục tiêu mong muốn hoàn thành. Trong phương diện này, phát triển phần mềm cũng như y học, không thể hoàn toàn dùng khoa học để giải quyết tất cả. Chỉ đáng tiếc là cho đến nay, những người thông thường vẫn không hiểu phát triển phần mềm là một nghệ thuật - Nghệ thuật phải có chuyên môn kỹ thuật.

Hiện nay, bạn phải biết sử dụng phương pháp của bác sỹ. Bác sỹ không thể nói rằng chữa trị sẽ đem lại kết quả như vậy, mà nói một cách đầy tự tin: "Hãy thử xem, tất cả đều chưa xác định được". Đối với quản trị dự án, thường khi không thể xác định là có thể làm hay không, dám đảm bảo tính năng,thời gian của sản phẩm. Càng nguy hiểm hơn, bác sỹ chỉ chữa trị một căn bệnh nhất định trong cả một hệ thống; Còn phát triển phần mềm lại phải đối mặt với hệ thống hoàn toàn mới mẻ, do đó tính bất định càng cao.

Đương nhiên, tình huống nào cũng có thể xảy ra, dù là hành vi đơn giản của bác sỹ, cũng có khả năng bị nguy hiểm.

Còn một sự việc khác cần phải học bác sỹ là, dù là những hành vi nhỏ và đơn giản đến đâu đều có chút nguy hiểm, do đó bác sỹ nói: "Đương nhiên, tình huống nào cũng có thể xảy ra, tỷ lệ chữa khỏi cao đến đâu cũng không đảm bảo 100% không có vấn đề gì". Nếu nói như vậy mà vẫn không làm cho các thành viên hiểu rằng sự việc nào trong tương lai cũng có tính bất định, đều có nguy hiểm thì tôi cũng đành bó tay.

Chúng ta cần phải ghi nhớ, đối mặt với tính bất định của phát triển phần mềm, vẫn có phương pháp để thử xem. Cũng như bác sỹ thử cho bệnh nhân hiểu hành vi chữa trị nào đều có chút nguy hiểm; Quản lý cũng phải làm cho thành viên hiểu, dự án nào cũng có tính bất định và nguy hiểm.

NGUYÊN TẮC 28: TAM GIÁC VÀNG PHÁT TRIỂN PHẦN MỀM: TÍNH NĂNG, TÀI NGUYÊN VÀ THỜI GIAN

Remember the triangle: features, resources, times

Là lãnh đạo phát triển phần mềm, bạn phải tập trung chú ý vào ba sự việc: Tài nguyên (người và tiền bạc), tính năng và thời gian. Đó là trọng tâm phát triển phần mềm, những cái khác đều là ngoại vi. Tài nguyên, tính năng và thời gian là ba mép của hình tam giác, thay đổi của mép nào cũng gây ảnh hưởng đến mép kia. Do đó, nếu thời gian tụt hậu, đi xem hình tam giác của bạn, xem ảnh hưởng đối với tính năng và tài nguyên; khi có người nói đến việc tăng tính năng nào đó, bạn phải nói đến thời gian và tài nguyên, thể hiện rõ phản ứng nhanh nhạy của tư duy. Vì vậy, nhiệm vụ đầu tiên của quản lý là để hình tam giác này trong lòng, sử dụng mô thức này bất cứ lúc nào để suy nghĩ vấn đề, bạn sẽ thấy đáp án đều ở trong hình tam giác. "Được thôi, thời gian của chúng tôi tụt hậu, phải sử dụng hai mép kia để bù đắp, hoặc điều chỉnh cả ba mép, có thể bớt chút tính năng, hoặc thêm chút tài nguyên, hoặc sửa đổi thời gian". Do con người, thời gian và tính năng đều là thứ bạn quan tâm, nên bạn có khái niệm về tam giác, chẳng mấy chốc bạn sẽ thấy hình tam giác hỗ trợ rất lớn đến suy nghĩ của bạn.

Còn có một điểm nữa cần nhớ, thời gian tụt hậu, bạn chỉ có bốn lựa chọn: Tăng thời gian, giảm tính năng, tăng vốn hoặc tiến hành cả ba. Nhưng con người không thể tăng một cách tuỳ tiện.

KHÔNG THỂ TĂNG THÊM NGƯỜI?

Bạn có nghe nói rằng, thêm người là một chiến lược sai lầm? Trong tác phẩm "Tài nguyên nhân lực" (The Mythical Man Month) đã có rất nhiều chương bàn về luận điểm: Trong dự án phát triển phần mềm, nếu tăng thêm người không hợp lý sẽ làm cho công việc càng chậm. Quan điểm này đã được chứng minh là đúng, nhưng chúng ta cần phải hiểu rằng đó chỉ là cuốn sách chứ không phải là luật pháp.

Nhưng, cách giáo dục truyền thống làm cho mọi người dùng phương thức lý giải quá đơn giản: không được thêm người vào dự án đã tiến hành được một nửa, trọng tâm là "Nhóm đang tiến hành". Điều này đã trở thành cái cớ cho các nhà quản lý quyết định không tăng thêm người. Đúng vậy, tăng thêm người sẽ khó quản lý (Phải quy hoạch trước khi bắt đầu dự án), đồng thời không phải biện pháp hay, chỉ suy nghĩ đến trong trường hợp bất đắc dĩ. Nhưng cũng không ai ngăn cấm điều này.

NGUYÊN TẮC 29: KHÔNG GIẢ VỜ HIỂU

Don't know what you don't know

Đối với những vấn đề không hiểu, không nên giả vờ hiểu, hoặc xem ra có vẻ hiểu, hoặc không muốn hiểu người khác mà giả vờ hiểu. Điều này sẽ gây khó khăn cho quản lý, nếu bạn phạm phải lỗi này, sẽ thấy sự cay đắng của nó. Mỗi người trong mỗi giai đoạn, chắc chắn có một số sự việc anh ta không biết, điều này hoàn toàn được phép. Che giấu những sự việc bạn không hiểu chỉ gây ra lệch lạc trong nhận thức của người khác, kết quả là do không biết các sự việc đó mà tin tưởng những thứ không đáng tin. Nếu bạn dũng cảm thừa nhận sự việc mình không hiểu, sẽ có thể vừa bỏ được lớp bùn này.

Mọi người đều thấy sự việc có tính quan trọng, mà mình lại không hiểu thì thật mất mặt, đó là tính cách bẩm sinh. Còn trong quá trình phát triển phần mềm, có rất nhiều điều mọi người không biết, do vậy quản lý hay nhân viên phát triển rất dễ có xu hướng không hiểu giả vờ hiểu. Nhóm phát triển tốt cần có một trang, trong đó liệt kê sự việc mà chúng ta không biết, như vậy mới có thể dễ dàng nắm được rốt cuộc chẳng có sự việc nào xác định cả. Để chống lại tâm lý kiêu ngạo "Tôi biết rồi" cần sự dũng cảm trong tâm lý của nhóm. Đối

với quản lý, khen ngợi thành viên dũng cảm thừa nhận không hiểu sẽ rất khó khăn, nhất là khi sự thực được che đậy. Che đậy giả mạo là tâm lý phòng ngừa sai lầm đối với những sự việc chưa biết.

Che giấu giả mạo là tâm lý phòng ngừa sai lầm đối với những sự việc chưa biết.

Tôi kiến nghị quản lý nên coi trọng thành viên hiểu được những chỗ nào mình không hiểu, chứ không phải bị ép hoặc tự nguyện che giấu. Cần cho nhóm của bạn biết trước đây họ không hề xem qua những thứ không biết của mình, nhưng bây giờ phải tập thừa nhận những việc mình không hiểu. Thông thường, vào thời kỳ đầu thành lập nhóm hay giai đoạn chuyển đổi rất dễ làm được. Cuối cùng, mọi người đều hiểu thành công rất quan trọng, hy sinh chút thể diện hoặc bị đả kích có quan hệ gì chứ?

Cần kiên trì ép thành viên phải đối mặt với tính bất định, thừa nhận mình không biết, sau đó chuyển không biết thành có biết. Nhiệm vụ của lãnh đạo là cho mọi người thấy tính bất định là sự thực tuyệt đối, yêu cầu mọi người phải đối mặt và thích ứng. Để có lợi cho mọi người, thành viên phải học cách sinh tồn trong môi trường bất định, đồng thời trở nên hứng khởi.

Chúng ta hãy xem quy định chính thức sau: Khi có một sự việc có tính bất định, đã trực tiếp nói lên sự thực này, cho dù sự việc có tính bất định là lúc nào có thể đưa ra phiên bản nâng cấp. Không phải lo lắng, không quản trị dự án nào nhận mình không hiểu mà bị mất ghế.

Không nói mọi người cũng hiểu, cấp dưới của tôi đều được cộng điểm khi biết mình không hiểu, bởi biết mình không hiểu thì mới có thể tìm kiếm tri thức. Tôi thà biết chỗ nào thiếu sót, như vậy mới biết việc gì quấn lấy mình, tốt nhất cấp dưới và đồng nghiệp cũng nên hiểu điều gì.

Mục tiêu của dự án phát triển phần mềm không phải là quy hoạch chính xác trước khi ra tay, mà hàng ngày khi chuyển từ sự việc chưa biết sang đã biết, đều có lựa chọn chính xác. Nếu rõ ràng bạn không hiểu chuyện gì đó mà lại giả vờ không biết, tức là bạn đã đưa ra quyết sách sai. Khi thông tin chứng minh là bạn sai, bạn sẽ cảm thấy khó xử. Do đó, tôi càng sợ thông tin, mà nhiều người cho rằng đang đối phó với sự thật, cuối cùng bạn rơi vào vòng tuần hoàn xấu.

Chỉ cần khi biết tính bất định ở đâu, bạn mới có thể giải quyết; Những sự việc xác định không được phát hiện cũng có thể bám lấy. Tóm lại, thừa nhận mình không hiểu còn hơn bị đánh gục.

Khi bạn không biết cần hoàn thành công việc như thế nào, khi có âm thanh nội tâm chất vấn bạn - hãy đối mặt với nó. Không phải sợ hãi thừa nhận mình không hiểu, thực tế là hoài nghi chưa xuất hiện của nhóm, mà bạn nhận được các tín hiệu vô hình. Có lẽ khi bạn nói với người khác, và nhận được câu trả lời: "Đúng vậy, tôi đang nghi ngờ về chuyện này". Đương nhiên, đôi khi người khác cho bạn là người vô lý, không sao, đó là cái giá nhỏ mà bạn phải chịu trong môi trường bất định này (Nếu thực tế mang cuộc sống quá thấp, có lẽ bạn nên xem bác sỹ tâm lý, nhưng tối thiểu cũng không làm bạn chậm quá).

ĐỒNG NHẤT TRONG NGOÀI

Đồng nhất trong ngoài là một loại cân bằng. Bạn có thể nói là làm? Hoặc nghĩ thế nào thì làm thế ấy? Hoặc suy nghĩ và hành vi trái ngược? Những gì thảo luận có thực là những điều họ muốn nói? Chúng ta có để người khác kéo mũi dắt đi? Hành vi, cảm nhận và mong muốn của chúng ta có đồng nhất? Khi chúng ta nói muốn làm, có thực sự muốn làm không? Chúng ta có thực sự đồng ý, hay giả vờ một chút cho họ tưởng rằng chúng ta đồng ý?

Tính đồng nhất dùng để phân biệt thành thực và giả mạo. Khi làm một việc gì đó, tôi làm do bản thân việc đó đáng làm hay nó có chút giá trị đối với người

đó? Tôi thành thực nói rõ tính cách nhìn nhận của bản thân, hay có xu hướng bảo vệ mình? Khi tôi cho rằng sự việc gì đó là ý tốt, tôi có đi làm không? Tôi có thực sự tin, hay là cho mọi người thấy rằng tôi tin?

Tính đồng nhất cũng chính là sự thực. Hiểu sự thực còn khó hơn là nói ra sự thực. Khi mọi người xung quanh đều cho một sự việc nào đó là đúng, nhưng tôi không thể chứng minh, chỉ là hoài nghi trực giác rằng điều đó là sai, và trước khi xây dựng mô hình suy nghĩ nhóm, tôi có nên nói ra cách nghĩ của mình, sau đó đi tìm sự thực?

NGUYÊN TẮC 30: XÂY DỰNG ĐIỂM KIỂM TRA THÍCH HỢP

Don't go dark

Giả sử khi gần đến ngày kiểm tra, bạn hỏi nhân viên phát triển: "Tình hình ra sao? Kịp tiến độ không?".

Anh ta trả lời: "Sáu tuần nay đều tiến hành thuận lợi, còn tiến độ hôm nay, là công việc lẽ ra phải làm xong từ 6 tháng trước".

Không, tụt hậu không chỉ trong một ngày, tụt hậu không phải chỉ đến khi gần kết thúc dự án mới phát hiện ra, cũng không phải phát sinh một ngày trước điểm kiểm tra, ngày nào nó cũng phát sinh, giờ nào cũng phát sinh. Mỗi khi bạn pha một tách cà phê, mỗi khi trả lời một email, mỗi khi lắp lại cái máy tính, bắt lỗi phần mềm, ... đều là thời khắc gây nên sự tụt hậu.

Để đối phó với vấn đề tụt hậu, bạn phải chia công việc phát triển thành nhiều đoạn nhỏ, mỗi đoạn nhỏ là một điểm kiểm tra, mỗi điểm kiểm tra đều phải xem kỹ xem có tụt hậu không? Chu kỳ khoảng cách giữa các điểm kiểm tra nên kéo dài bao lâu? Nếu lâu quá, hiệu quả không nhiều; Nếu ngắn quá, chỉ sợ khó có thể chia nhỏ công việc như vậy. Phần mềm không đơn giản như một đống gỗ, quan hệ giữa chương trình là thể động và

lập thể. Trong nhóm của tôi thường tranh luận không ngớt: 5 ngày? 10 ngày? 3 tuần? Theo kinh nghiệm của tôi, ba tuần đủ để tình hình mất khả năng khống chế.

Chu kỳ kiểm tra thích hợp của mỗi công ty một khác. Cách làm của chúng tôi là, để các thành viên trao đổi, trong thời gian nhất định phải làm xong công việc của mình để trao cho người khác, nếu không anh ta không thể kịp thời tiến hành, như vậy sẽ tạo nên vòng cắt công việc thích hợp, hơn nữa nếu kéo dài là sẽ nhận ra ngay. Ví dụ, chúng ta biết tuần này bị tụt hậu một ngày, một ngày có thể không phải chuyện nhỏ, điều này rất phải nói rõ, nó tốt hơn việc mỗi năm sau mới biết mình tụt hậu, chỉ sợ khi tụt hậu đến mức đấy thì dù máy tính có lạc hậu đến đâu cũng không kịp nữa rồi, các vấn đề đáng tiếc trên thế giới thường như vậy.

Vấn đề đáng buồn trên thế giới, là bạn lạc đường trong phát triển phần mềm. Trong giai đoạn này có thể bạn học được kinh nghiệm hoàn toàn bị nhầm lẫn này: "Tôi không biết phải làm gì mới đúng, tôi không biết hiện đang cách mục tiêu gần hay xa, tôi chỉ biết tình hình bây giờ rất tồi tệ, biện pháp nào tôi đưa ra cũng làm tôi rối rắm thêm, tôi chỉ xác định được là làm gì cũng hoảng..." cảm giác đáng sợ này bắt nguồn từ hộp đen - công việc được tiến hành không có người kiểm tra. Hãy bật đèn lên, cho thành viên vào làm trong môi

trường rõ ràng minh bạch, ngày nào kết thúc công việc của ngày.

LẠC ĐƯỜNG TRONG PHẦN MỀM

Tôi không cần mất nhiều thời gian bàn về vấn đề nghiêm trọng do tụt hậu trong phát triển phần mềm, bạn cũng hoàn toàn hiểu rõ, nếu không bạn sẽ không thấy đáp án trong cuốn sách này. Gọi bất kỳ một nhân viên phát triển hay phụ trách dự án tới hỏi, họ đã có kinh nghiệm về tụt hậu tiến độ? Đáp án chỉ có thể; Một là có; Hai là trên bề mặt là không nhưng thực tế là có. Trên bề mặt là không nhưng đó có thể tăng cường tiến độ, hoặc chấp nhận giảm thấp mục tiêu.

Vấn đề đau đầu hơn là, có bao nhiêu quản trị dự án và nhân viên phát triển cảm nhận được mình đang lạc đường.

Đã là người làm trong ngành phần mềm, hầu hết đều có thể hiểu được hàm nghĩa của vấn đề này, lạc đường là cảm giác rất đáng sợ. Có rất nhiều lý luận giúp bạn tính được chỉ tiêu tiến độ, nhưng không thể giúp bạn thoát khỏi buồn phiền, ai cũng cảm thấy không vui, bạn bắt đầu mất đi khả năng phán đoán nhanh nhạy, ngay cả những dự án dở hơi cũng được xem xét cẩn thận. Lo lắng, sợ hãi, buồn phiền và lại thấy có chút hy vọng. Về căn bản bạn không biết mình

đang làm gì, sự việc được làm ra sao, vấn đề rốt cuộc như thế nào, động tác nào của bạn cũng gây ra thất bại lớn, bạn hoàn toàn chịu bó tay.

Thế là, bạn dần có một trực giác tản mát, nhất định mình không có tài năng mới ra như vậy. Tiếp đó, ai cũng thấy sự sợ hãi của bạn, đồng thời mất niềm tin với bạn; Thành viên không còn tin cậy bạn, bạn còn muốn lãnh đạo họ ra sao đây?

Những điều này chính là nỗi đau của mỗi quản trị dự án.

Hiện nay, dựa vào tính quan trọng của điểm kiểm tra, bạn nói với nhân viên phát triển: "Chúng ta cần kiểm tra tiến độ hàng tuần". Có người phản ảnh kiểu quản lý như vậy quá chi tiết. Để những người này có thể chấp nhận phương thức quản lý này, bạn phải nói rõ cho họ, còn rất nhiều người đang đợi phần mềm này ra đời mới làm được việc - còn phải viết tài liệu, đánh giá chất lượng, và còn nhiều nhân viên phát triển khác cũng muốn biết kết quả của giai đoạn này, mới viết tiếp hoặc kiểm tra các chương trình liên quan. Lịch trình làm việc giữa các thành viên có quan hệ trên dưới và phản hồi, cùng dựa vào tiến độ của nhau, từ đó hình thành nên động cơ của điểm kiểm tra, chứ không dựa vào cưỡng chế cấp trên.

Một vài tính năng sản phẩm phải mất tới vài tháng hay vài năm mới hoàn thành, nhưng việc nắm lịch trình không liên quan đến nhiệm vụ ít nhiều, bởi bất cứ một tiến độ nào tụt hậu đều dần được tích tụ lại. Cũng tức là chia cắt công việc phải đủ chi tiết, một khi một đoạn bị lỗi, có thể đuổi kịp trong thời gian ngắn nhất. Trên thực tế, chu kỳ kiểm tra, một tuần đã là đủ dài rồi, nghĩ mà xem, nếu muốn đuổi kịp tiến độ vào tuần sau, bạn có làm được không. Do đó, quản lý chặt làm nhân viên không thích, nhưng để hoàn thành sản phẩm đúng tiến độ không thể không làm như vậy; Hơn nữa sau đó ai cũng coi lịch trình là quan trọng nhất, lúc đó mọi người đều cảm thấy bảng lịch trình tạo cảm giác thành công, và rất vui vẻ đi làm.

Ngoài ra, dựa vào tác dụng tương hỗ và đảm bảo tiến độ giữa các đồng nghiệp, cũng là một trong những nhân tố của điểm kiểm ra thuận lợi. Cũng như mục đích thiết kế sản phẩm làm cho các thành viên tham gia, ai cũng đóng góp biện pháp tốt nhất của mình. Thực hiện điểm kiểm tra cũng như vậy, mỗi người thực hiện cam kết của mình, chế độ như vậy sẽ thành công. Ví dụ, nhân viên phát triển bảo rằng thứ 6 sẽ chuyển chương trình cho nhân viên tài liệu, lời hứa đó trở thành một phần của công việc, anh ta sẽ cố gắng hoàn thành đúng hạn; Mặt khác, khi quản lý ra lệnh cho nhân viên phát triển hoàn thành vào tháng 6, nhưng

uy quyền của quản lý tạo ra phản cảm và trở thành một phần của công việc, nhân viên có cảm giác không yên tâm. Đây cũng là hiện tượng thú vị trong quá trình khai thác: Trạng thái tâm lý của một người có ảnh hưởng trực tiếp đến kết quả.

Phía trước tôi đã nhắc đến sự đồng nhất, ảnh hưởng đối với phát triển phần mềm chính là vấn đề tiến độ, ai cũng coi trọng lời hứa của mình, không lừa dối nhau, việc thực hiện điểm kiểm tra cũng rất thuận lợi. Đối với hàng ngàn đoạn công việc, chỉ có thể duy trì dựa vào cam đoan như vậy. Điều này không liên qua đến mục tiêu chỉnh thể, nhưng lại liên quan đến việc dũng cảm đối mặt với dũng cảm đồng nhất.

NGUYÊN TẮC 31: CẨN THẬN KHÔNG CÓ THÀNH VIÊN Ở ĐIỂM KIỂM TRA

Beware of a guy in a room

Nguyên tắc 31 là trường hợp đặc biệt của nguyên tắc 30, nhưng cần phải nói riêng.

Đã có lần (Thực ra ba lần, chúng tôi không dám kể cả ba câu chuyện ra), tôi giao công việc khó khăn nhất cho người xuất sắc nhất, có sức sáng tạo nhất và năng lực kỹ thuật tốt nhất, chúng tôi có được anh ta là may mắn lớn. Không ai hoài nghi khả năng và sức phán đoán của anh ta.

Nhưng khi bắt đầu công việc có tính thách thức đó, anh về phòng đóng cửa, lúc này xung quanh anh chỉ còn âm nhạc, kẹo và tạp chí chiến tranh giữa các vì sao, cũng như 6 hộp nước hoa quả, sau khi nghỉ ngơi thư giãn, anh bắt đầu làm việc. Các thành viên khác đều tin tưởng vào anh, thật may mắn khi chúng tôi có anh để giải quyết nhiệm vụ nặng nề nhất. Bên ngoài chúng tôi nghe thấy tiếng gõ bàn phím không ngừng nghỉ, trong lòng thầm vui mừng, tuy không ai biết chương trình trong phòng như thế nào, nhưng chúng tôi đều cảm thấy một cao thủ đang nỗ lực cho công ty là tốt lắm rồi.

Người anh ta vã mồ hôi, chắc chắn là áp lực rất lớn.

(Đây là tình hình lạc quan chưa từng có, mục tiêu của chúng tôi rất rõ ràng, chúng tôi có nhóm nhân viên ưu tú, tuy đôi lúc vẫn ngồi nhâm nhi cà phê, nhưng chúng tôi đều tin tưởng mình đang có một kế hoạch lớn, người khác phê bình chúng tôi liền bịt tai không nghe).

Dần dần, chúng tôi bắt đầu nhận thấy anh ta gặp khó khăn. Dần dần không thấy anh về nhà, âm nhạc cũng ngừng, người anh bắt đầu vã mồ hôi , có vẻ như... tiến độ có vấn đề.

Toàn bộ công việc chỉ có mình anh làm, không có điểm kiểm tra, chỉ cần lúc hoàn thành anh giao sản phẩm là được, đương nhiên những người khác đã hoàn thành công việc của mình đúng hạn, chúng tôi đành phải gõ cửa: "Uylliam, làm thế nào rồi?" phía trong không có động tĩnh, một lúc sau anh ta nói "Tạm ổn", "Lúc nào có thể hoàn thành?" Một lúc sau, chúng tôi có vẻ nghe thấy tiếng khóc, anh ta nói một cách nặng nhọc: "Cũng nhanh thôi".

Chúng tôi biết từ "nhanh" của anh, thực ra còn sớm hơn nhiều. Hiện nay cả công ty chỉ ngồi chờ, không thể giúp đỡ được gì. Còn tôi, với tư cách là phụ trách dự

án, đối mặt với khó khăn này: Tôi có thể loại bỏ Uylliam, nhưng nghĩ đến anh là người duy nhất có thể cứu chúng tôi, duy nhất có thể hiểu được khó khăn trước mắt, do đó tôi không suy nghĩ gì thêm mà quyết định để anh tiếp tục hoàn thành công việc, dẫu sao anh ta cũng là người ưu tú nhất, đồng thời là người duy nhất nắm vững vấn đề này.

Tôi chỉ làm được có một vấn đề, đó là mua hộ anh Côcacôla.

Một phương pháp có vẻ tốt nữa là tăng thêm chút áp lực, làm cho anh thấy rõ mình đang giữ sự sinh tồn của nhóm, các thành viên và gia đình của họ đều hy vọng vào bộ óc thông minh và ngón tay gõ nhanh thoăn thoắt của anh, chúng tôi cần phải thúc anh hoàn thành.

Nhưng tôi cũng hơi băn khoăn, cảm thấy đây không phải là biện pháp tốt, không giúp gì về sức sản xuất. Tôi không có cách gì để thoát khỏi tình hình này, tôi chỉ có thể suy nghĩ tích cực: Uylliam có thể kết thúc công việc, chỉ mong anh không chết hay từ chức. Qua bài học này, có lẽ anh ta không dám làm cho chúng tôi nữa. Việc làm duy nhất của tôi là giúp anh ta mua Côcacôla, ngoài ra chẳng còn cách gì khác!

Đương nhiên, nếu bạn có một nhân viên xuất sắc đóng cửa vùi đầu làm việc, bạn có lẽ sẽ có kết quả khá

đặc biệt: Một là không đưa ra được sản phẩm ở các điểm kiểm tra, nhưng vào 5 phút cuối của thời hạn đưa ra một kiệt tác; Hai là tránh mọi phiền nhiễu ở đời, toàn tâm toàn ý làm việc, do mức sáng tạo của công việc quá cao, hoàn toàn phải dựa vào tâm huyết của anh ta, lúc này hợp tác nhóm không có ích gì mà có khi còn quấy rối.

Loại lập trình viên thứ hai này, để yên tĩnh nên đóng kín cửa ở trong phòng một thời gian dài, không ai biết tiến triển công việc như thế nào, có lẽ do yêu cầu phát huy sức sáng tạo, nhưng sẽ rất nguy hiểm đối với việc đưa ra sản phẩm đúng hạn, dù có xuất sắc đến đâu thì lãnh đạo cũng không nên giao công việc quan trọng cho anh ta, trừ phi làm cùng với các nhân viên phát triển khác định kỳ kiểm tra tiến độ công việc, hơn nữa có yêu cầu khắt khe về thời hạn. Đáng tiếc là có một số nhân tài không chấp nhận được những hạn chế này. Trong ngành phần mềm, cũng có những người như vậy, họ có thể làm ra những tác phẩm không thể tưởng tượng nổi, đưa kỹ thuật phát triển, nhưng họ khó có thể có mặt trong nhóm phát triển phải đưa sản phẩm ra đúng hạn.

Dù là loại nào, cho phép nhân viên phát triển đóng cửa phòng vào làm việc không ai biết tiến độ công

việc như thế nào, kết quả thường ảnh hưởng xấu đến cả nhóm. Do đó, phải cố gắng tránh tình hình này, tuyệt đối không cho phép "Thành viên không có điểm kiểm tra", nếu không sẽ khó tránh khỏi thất bại.

NGUYÊN TẮC 32: THƯỜNG XUYÊN TÍCH HỢP, PHẦN MỀM SẼ THUẬN LỢI RA ĐỜI

If you build it, it will ship

Tích hợp các chương trình thành phần mềm mới có thể tung ra thị trường, điều này rất tự nhiên, không thể bán cho khách hàng đống mã chương trình không chạy được! Nhưng ý của tôi là nếu có thể thường xuyên và định kỳ tích hợp phần mềm thì tốt hơn là đến ngày cuối cùng mới kết hợp một lần. Bạn phải luôn chuẩn bị chỉnh thể hiện trạng phần mềm để cho mọi người xem mới được.

Bạn thử tưởng tượng một trăm thợ xây nhắm mắt để xây một toà nhà xem! Không biết mình đang xây phần nào của toà nhà, không biết xây đến mấy tầng, mà chỉ quan tâm đến phần của mình. Toà nhà như vậy có xây được không? Nếu có xây xong thì cũng chắc chắn sẽ đổ sập! Do đó, thường xuyên tích hợp phần mềm là điều rất cần thiết, trọng điểm của nguyên tắc này không chỉ cần thường xuyên và định kỳ tích hợp phần mềm trong cả quá trình phát triển phần mềm, mà còn cố gắng kết hợp nên phần mềm hoàn chỉnh và chính xác nhất, quan trọng hơn kết quả xong phải đặt tại nơi công khai để mọi người đều thấy.

Đã có trường hợp, các chương trình đều được viết theo đúng tiến độ, các kiểm tra cá biệt cũng đã hoàn thành, nhưng sau đó không thể tích hợp lại thành phần mềm hoàn chỉnh. Do giữa các chương trình phải liên kết lẫn nhau, nhưng ngược lại có chương trình lại hạn chế chương trình khác. Trong quá trình viết chương trình phải luôn chú ý xem chương trình của mình viết có phối hợp tốt với các chương trình khác không, cách tốt nhất là phải thường xuyên tích hợp.

Phần mềm không nhất thiết ngày nào cũng kết hợp, nhưng phải thường xuyên và định kỳ làm, không chỉ các chương trình cần kết hợp mà bao gồm cả phần cài đặt và hỗ trợ trực tuyến sau đó công khai cho mọi người xem, để nhân viên bảo đảm chất lượng có thể đánh giá hiện trạng phần mềm, cũng như quan sát tình hình phát triển hay đang tụt hậu. Tích hợp phần mềm theo quy luật là một chỉ tiêu rất đáng tin, biểu thị nhóm có vận hành bình thường không, phần mềm có hoàn thành không.

Tích hợp phần mềm cần có thời gian, chương trình cũng rất phức tạp, đáng để bạn nghiên cứu chiến lược tốt nhất cho nó. Trong nội bộ Microsoft có rất nhiều nhóm phát triển sử dụng chiến lược tích hợp hàng ngày, đồng thời còn có nhóm chuyên trách (Build master). Kết hợp chương trình rất quan trọng, không bị

thất bại hay đứt quãng. Khi đưa một chương trình vào (Check-in) môi trường phát triển, sẽ phải phối hợp với chương trình khác, nên nếu đoạn chương trình nào đó có vấn đề, sẽ dễ dàng tìm được người phụ trách, do đó cảm giác trách nhiệm và danh dự của thành viên là mình phải làm ra chương trình tối thiểu đạt tiêu chuẩn, và thuận lợi trong việc tích hợp hàng ngày. Đương nhiên bạn có thể nghiêm khắc hơn, theo kinh nghiệm của tôi, kết hợp mỗi ngày một lần phương thức hữu hiệu nhất (Xem thêm nguyên tắc phía sau).

Mặt khác, phụ trách dự án cần đưa ra chiến lược tích hợp tốt nhất, chứ không phải vài ngày kết hợp một lần có tính tuyệt đối. Không cần thiết phải ngày nào cũng làm, cần tìm ra khoảng cách tốt nhất và ngắn nhất, với trọng điểm là kết hợp được phần mềm với khoảng cách ngắn nhất.

Nói thực, tôi không biết phần mềm "cần" bao nhiêu chu kỳ tích hợp là tốt nhất, nhưng tôi tin rằng hầu hết các công ty phần mềm đều làm chưa đủ về mặt này. Đối với một số nhóm phát triển, mỗi ngày tích hợp sẽ hơi nhiều, nhưng tôi đảm bảo đối với bất cứ nhóm phát triển nào, mỗi tuần tích hợp một lần không phải là ít nhất.

Phụ trách dự án phải thường xuyên để ý đến tình hình chỉnh thể của phần mềm, mới có thể biết được

đang làm gì. Dù phần công công việc và thiết lập điểm kiểm tra rất tốt, sự tin tưởng và quan hệ giữa các thành viên cũng khá lý tưởng, nhưng nếu không thể Tích hợp phần mềm thì bạn chỉ nắm được tiến độ trogn tưởng tượng và dự đoán thôi, chứ không căn cứ vào thực tế, do vậy bạn vẫn chưa thực sự nắm được tình hình phát triển phần mềm.

Mỗi lần bạn tích hợp phần mềm, đều có thể tính ra được bao nhiêu lỗi, đồng thời nắm vững tiến độ phát triển phần mềm, bạn cũng nhìn thấy các tính năng đang thành hình.

Để có thể tích hợp được phần mềm, các chương trình phải duy trì trong tiêu chuẩn chất lượng nhất định, các lỗi lớn sẽ được phát hiện kịp thời. Do đó, vào mỗi thời gian kết cấu phần mềm cố định, mới có thể duy trì phần mềm có thứ tự và chất lượng nhất định, điều này chắc mọi người đều cần biết.

Quan trọng hơn, định kỳ tích hợp phần mềm cũng như nhịp tim đập của nhóm vậy, buổi chiều mỗi ngày mọi người đều nhìn thấy thành quả công việc của cả nhóm trong ngày, mỗi ngày đều thấy sự tiến bộ, nhóm sẽ được cổ vũ rất lớn. Nếu có ngày phần mềm kết hợp thất bại, hoặc vì lý do nào đó mà không kết hợp sẽ ngay lập tức làm cho mọi người chú ý và căng thẳng, nhất

định phải tìm nguyên nhân để sửa đổi. Nếu kết hợp thường xuyên thất bại, sẽ là điềm báo cả dự án thất bại.

Nhưng đừng có làm lộn lên, tích hợp phần mềm mà tôi nói tới, không phải là kết hợp quy mô nhỏ của lập trình viên trên máy tính của mình, mà là kết hợp chính thức và công khai, kết hợp các chương trình lớn nhỏ lại thành một phần mềm mà ai cũng có thể nhìn thấy và sử dụng. Nhân viên bảo đảm chất lượng có thể kiểm thử sơ bộ, xem chức năng cơ bản của nó có sai không. Nếu ngày nào cũng Tích hợp phần mềm, tối thiểu bạn cũng biết được mình không thất bại, mọi người cũng thấy nó đang tiến tới mục tiêu, tính năng dần dần hình thành; Hơn nữa mọi người kiểm thử phần mềm sẽ có nhận thức tương đồng về hiện trạng của phần mềm.

Thường xuyên tích hợp phần mềm còn có các lợi ích sau:

- Thường xuyên và công khai tích hợp phần mềm có thể thấy được độ tín nhiệm thực sự giữa các thành viên. Chỉ cần bất cứ một mắt xích nào không chặt, phần mềm sẽ không được kết hợp, quá trình xem xét kết hợp cũng như xem xét quan hệ giữa các thành viên hay quan hệ đối ngoại của nhóm, nếu có vấn đề ở nơi nào đều sẽ tìm thấy một cách dễ dàng.

- Phần mềm tích hợp ra có thể bộc lộ những vấn đề không được chú ý đến khi thiết kế, như hiệu quả sử dụng, kích thước đối tượng, ... Nếu vấn đề này phát hiện muộn quá sẽ không kịp sửa đổi.

- Tích hợp phần mềm làm bước đi của các thành viên được đồng nhất. Thông thường, vấn đề hay gặp phải trong tích hợp phần mềm là hài hoà giữa các phiên bản, nghiêm trọng hơn là mỗi người có một phiên bản riêng, không ai biết người người khác đang làm phiên bản gì, khi tích hợp phần mềm công khai, các phiên bản sẽ đồng bộ.

- Tích hợp phần mềm có thể làm cho các thành viên đối mặt với các vấn đề mà họ bỏ qua. Nhóm tương đương với phần mềm, do đó trạng thái phần mềm hiện tại chính là trạng thái của nhóm. Khi tôi thấy sản phẩm có vấn đề, tôi truy hỏi thành viên: "Tình hình tích hợp phần mềm như thế nào?" Rất nhiều kiểu trả lời, nhưng chỉ có một ý: "Chúng tôi mất rất nhiều thời gian mới kết hợp thành công một lần, có lúc phải mất hai tuần, chúng tôi hy vọng có thể rút ngắn thời gian, nhưng do một số nguyên nhân nào đó, chúng tôi không thể kết hợp theo kế hoạch".

Đúng vậy, muốn thường xuyên và định kỳ kết cấu phần mềm sẽ gặp phải rất nhiều khó khăn, nhưng giải quyết được những khó khăn này sẽ giúp bạn có được

nhóm lành mạnh và phát triển phần mềm thuận lợi. Nhân viên phát triển phải tự kiểm tra kỹ các chức năng xem đã hoàn thiện chưa mới đưa ra kết hợp, chương trình không chỉ phải chạy chính xác về lôgic, mà còn phải chú ý đến *tài nguyên chiếm dụng* (Dùng xong bộ nhớ thì phải giải phóng), chạy có hiệu quả. Nếu các chương trình kết hợp đều tốt, sẽ rất dễ kết hợp thành phần mềm, đồng thời giảm bớt hao tốn thời gian do phải kiểm tra lại.

Trong quá trình phát triển, rất dễ có cảm giác sai về trạng thái của phần mềm, nhưng nếu có thể tích hợp phần mềm hàng ngày, những điều bạn nhìn thấy sẽ cụ thể hơn.

NGUYÊN TẮC 33: NẮM VỮNG TÌNH HÌNH THỰC TẾ

Get a known state and stay there

Đây là một đạo lý đơn giản mà hữu ích, ý nghĩa rộng hơn của nó là "Ngày nào cũng phải có sản phẩm ra đời".

Bạn phải hiểu rất rõ ràng về tình hình thực tế của phần mềm, nhất là khi chuẩn bị tung ra, bạn phải biết nó có hình thức, kết cấu, hay tính năng gì. Khi bạn coi nó như sản phẩm tung ra thị trường, bạn phải biết rõ tình hình có thể biết.

Nếu bạn hỏi nhân viên phát triển về tình hình phần mềm, anh ta có thể trả lời, nhưng không thể phản ánh đúng.

Giả sử có người yêu cầu bạn đưa thêm một tính năng nào đó vào sản phẩm mới vừa ra mắt, bạn có thể hiểu ngay được ý nghĩa của tính năng này đối với sản phẩm, đồng thời biết nên làm gì để đưa tính năng này vào, có gây xung đột lớn không, ... Bạn có thể triệu tập một số nhân viên phát triển, nhân viên bảo đảm chất lượng để thảo luận là có thể biết được nhận thức của

mọi người về tính năng này, mọi người đều rất rõ về vai trò và phạm vi nhiệm vụ của mình, sau đó bạn có thể nói: "Nào, bắt tay đi thôi!"

Có lẽ vài ngày sau, có lẽ khoảng một tuần, vài thành viên cầm đĩa mềm đến tìm bạn, cho bạn biết tính năng mới đã được làm tốt, chỉ cần cài đặt là đưa tính năng này vào. Thật không có khó khăn gì, phải vậy không?

Đó là do bạn có thể nắm rõ tình hình. Bạn cần biết, đưa ra một phiên bản mới cũng như nhiều lần không ngừng đưa thêm tính mới, sau đó đưa ra. Đây là phương thức để bạn hiểu được hoạt động phát triển phần mềm của nhóm! Lấy một số người có nhiệm vụ khác nhau vào một nhóm, phụ trách về một tính năng sản phẩm; Mấu chốt là bạn phải luôn nắm vững phần mềm và trạng thái đưa ra thị trường. Không được để phần mềm trở thành một đống mã hỗn độn. Không được để tình trạng bạn không hiểu gì về phần mềm, cần phải nắm chặt lấy nó không buông.

Muốn nắm tình hình của phần mềm, cần có nhân viên bảo đảm chất lượng giúp bạn, bạn cần có người chuyên kiểm thử tình trạng phần mềm. Nếu bạn hỏi nhân viên phát triển, có thể anh ta trả lời đúng, nhưng có thể là chẳng may đoán trúng. Tuy nhân viên phần mềm là người trực tiếp sáng tạo phần mềm, nhưng họ

không thể biết được bộ mặt thật mang tính toàn diện của phần mềm, điều này còn phải được nhân viên đảm bảo chất lượng đánh giá, để chuyên gia ngoài cuộc cho bạn biết tình hình phần mềm.

Hơn nữa, hàng ngày bạn đều nên tự làm kiểm tra nhỏ, một phần mềm là tự động, một phần làm thủ công; Cứ một hay hai tuần *kiểm thử* toàn bộ một lần, đảm bảo phần mềm ở trạng thái sẵn sàng tung ra thị trường.

Như thế nào mới được gọi là "Nắm vững tình hình phần mềm"? Chính là vào một điểm thời gian nào đó, đều có thông tin chính xác về mọi trạng thái của phần mềm. Do nhân viên đảm bảo chất lượng đã kiểm tra các chương trình hay bộ phận nên bạn có thể yên tâm thông tin là chính xác. Trên tay bạn phải có một tờ giấy, ghi rõ các tính năng đạt hay không đạt, số lỗi, tỷ lệ phát hiện lỗi và tỷ lệ loại bỏ lỗi và một số dữ liệu quan trọng khác.

Mục tiêu dự kiến được hoàn thành dù có thể đưa sản phẩm ra thị trường.

Nhưng không phải nhân viên đảm bảo chất lượng đánh giá quyết định có thể đưa ra được hay chưa. Do phát triển là nhiệm vụ của cả nhóm, phần mềm có ra

được không thì trong lòng ai cũng biết, điều này không nên có tranh luận khác - mục tiêu thiết kế thực sự hoàn thành, sản phẩm sẽ ra đời. Khi bạn đã đánh dấu hoàn thành lên các chức năng dự định, bạn sẽ có cảm giác thành công thực sự (Còn có thù lao của bạn nữa).

Nhưng điều kiện tiên quyết là bạn có thể nắm vững tình hình của phần mềm, đồng thời có được những thông tin mới nhất. Bạn phải biết cách dùng nhân viên bảo đảm chất lượng làm được như vậy.

Lúc nào cũng nắm vững tình hình phần mềm thực ra là rất khó, bạn cần có một nhóm nhân viên đảm bảo chất lượng rất ưu tú. Rất nhiều công ty phần mềm không có hoặc rất ít nhân viên đảm bảo chất lượng, do đó không thể nắm được tình hình phần mềm. Trong thực tế, nhóm phát triển phần mềm hiện đại không thể không có nhân viên đảm bảo chất lượng.

CỘT MỐC GHI CHÉP

Đoạn này chủ yếu nói rõ thêm nguyên tắc 33. Tôi muốn giải thích tường tận về các quan niệm này trong phát triển phần mềm, tất cả đều là những trăn trở, suy nghĩ, thu thập của tôi trong nhiều năm, được tích lũy nhờ bản thân và các đồng nghiệp tài giỏi đã trải qua không ít sai lầm mà có.

Bạn muốn có chỗ đứng trong ngành phần mềm? Bạn muốn có những bước tiến vượt trội? Vì vậy bạn cần phải có cột mốc đánh dấu (Milestone), biểu thị mục tiêu mà bạn đang đạt được, nhưng bạn phải tiến đến cột mốc tiếp theo.

Cột mốc phải có tiêu chuẩn đo đạc nhất định, nếu không sẽ không đạt tới, do đó không được thiết lập một mục tiêu không thể xác định rõ. Mỗi mục tiêu, dù lớn hay bé, đều cần được ghi chép vào một hồ sơ riêng, tài nguyên không thể đầu tư một cách tùy tiện, mơ hồ.

NGUYÊN TẮC 34: CỘT MỐC KHÔNG HỤT

Use ZD milestones

Trong phát triển, bản chất phần mềm là vô hình. không thể nắm bắt. Một phần vẫn tồn tại trong đầu óc nhân viên phát triển, một phần ở trong các dãy số không nhìn thấy của phần mềm, một phần nằm trong các giấy tờ của bản kế hoạch, một phần là những ý thức tiềm ẩn, luôn luôn biến đổi, chỉ được kích hoạt trong một cơ hội thích hợp. Mà ý thức tiềm ẩn của nhóm đều biểu hiện trong hoạt động thường ngày hay các quyết định của nhóm. Như phần trước đã nói, phần mềm tương đương với nhóm, do đó mọi tình hình của nhóm đều biểu hiện trong phần mềm, phần mềm chính là hình ảnh sống của nhóm.

Điểm không hụt không có nghĩa là phần mềm không có lỗi, cũng không có nghĩa là đầy đủ các chức năng.

Nếu muốn quản lý hoạt động phát triển phần mềm một cách hợp lý, cần phải định lý kết hợp, lúc đó mọi nỗ lực vô hình hay hữu hình của mọi người đều có thể được nhìn thấy, đồng thời cũng phân tích được ý

thức tiềm ẩn. Khi kết hợp các hình thức trong các chương trình phân tán thành một chỉnh thể, bạn có thể lấy được các thông tin hoàn chỉnh về tình hình phần mềm, xem cần phải tiến hành những gì. Nhưng dù thường xuyên tích hợp có rất nhiều điểm tốt, nhưng cũng cần rất nhiều thứ - bao gồm thiết kế phát triển phần mềm và các động lực tiềm ẩn của nhóm, thực tế đó là những công việc rất phức tạp.

Tung sản phẩm ra chính là cột mốc cuối cùng

Tạm thời không quản phức tạp đến đâu, để công việc phát triển phần mềm diễn ra thuận lợi, bạn tối thiểu phải để thành viên coi công việc là một gạch thẳng, có dũng khí đối mặt với khó khăn, đồng thời tăng cường hiểu biết của thành viên với hiện trạng phần mềm, để họ có niềm tin và năng lực phán đoán xem mình cần bỏ ra bao nhiêu công sức để có kết quả như vậy. Lúc đầu mọi người đều rất đau khổ, nhưng dần dần sẽ có các năng lực này, sẽ đối phó một cách tự nhiên, công việc phát triển cũng thuận lợi hơn (Xem nguyên tắc 3).

Rất nhiều nhóm phát triển trong Microsoft sử dụng phương thức "Cột mốc không hụt" (Zero Defects Milestone) để hiểu về tình hình phần mềm. Điểm

không hụt không có nghĩa là phần mềm không có lỗi, cũng không có nghĩa là đầy đủ các chức năng, mà chỉ thành phẩm của nhóm đạt đến tiêu chuẩn chất lượng theo kế hoạch, cũng đã được kiểm thử, tức là cột mốc không hụt. Trong nhóm Visual C++, mỗi lần tung sản phẩm ra đều quy hoạch trước ba hoặc bốn cột mốc, một chu kỳ sản phẩm thường là khoảng 1 năm.

Đương nhiên, cột mốc cuối cùng cũng là cột mốc cao nhất, là việc tung sản phẩm ra, nhưng các cột mốc khác cũng rất quan trọng.

Trong nguyên tắc 33 chúng ta khởi xướng, phần mềm phải luôn ở trạng thái tung ra, mà ngày nào cũng vậy, luôn luôn như vậy, đây là một lý tưởng nhưng trong thực tế dù không có hiện tượng tiến độ bị tụt hậu, cũng có lúc không thực hiện được. Bởi ngẫu nhiên bạn cần quay lại để xử lý các vấn đề trước mặt, cũng như tạm thời loại bỏ vật thay thế để đổi mới chức năng (Như trong quá trình xây đường cầu mở tạm một con đường nhỏ nhằm đảm bảo giao thông suốt, làm xong đường thì phải loại bỏ con đường nhỏ). Còn cột mốc không hụt, lại là một điểm sau một thời gian khá dài, nhằm đảm bảo trạng thái phần mềm đạt được tiêu chuẩn chất lượng như dự kiến, gần như định kỳ quét dọn hay kiểm tra, không để các lỗi nhỏ tồn tại.

Một cột mốc thông thường từ 6 tuần đến 2 tháng, khi đến ngày này, ngoài việc đạt được mục tiêu cột mốc, tất cả các hoạt động phát triển đều không được vượt cột mốc này. Do đó, mục tiêu cột mốc thường được gọi là "Sản phẩm trung gian" mà mọi người đều nhìn thấy. Sau khi đạt được mục tiêu cột mốc, cột mốc mới lại bắt đầu.

Nguyên tắc này nói thì dễ làm thì khó, nếu bạn có nhóm phát triển được trao đủ quyền, các thành viên có thể cùng xác định mục tiêu cột mốc, quản lý sẽ không phải lo lắng gì. Mỗi mốc mục tiêu đều được nhân viên đảm bảo chất lượng xây dựng quy tắc kiểm nghiệm trước, liên kết các chi tiết tiêu chuẩn chất lượng sau kiểm nghiệm, hơn nữa mọi người đều đồng ý và có nhận thức chung. Cũng chính là nói, tiêu chuẩn chất lượng sản phẩm trung gian do thành viên bàn bạc mà thành, chứ không phải là mệnh lệnh của cấp trên. Mỗi sản phẩm trung gian đều có quy tắc kiểm nghiệm trước, nói rõ mỗi quy tắc cần được thông qua vào lúc nào, các quy tắc kiểm nghiệm này sẽ được quản trị dự án tập hợp lại, đó chính là cột mốc.

Điểm tốt lớn nhất của "Cột mốc không hụt" là mỗi khi tiến độ tụt hậu, có thể phát hiện ngay và sửa đổi trong thời gian ngắn nhất, đồng thời đảm bảo vấn đề được tối thiểu hoá, xử lý kịp thời. Nếu mỗi một hoặc hai

tháng bạn có một cột mốc, bạn sẽ khống chế được tiến độ trong các cột mốc này. Tiến độ tụt hậu trong một cột mốc luôn ít hơn so với cả dự án, và cũng dễ đuổi kịp, vẫn còn tốt hơn là đến khi chuẩn bị tung ra mới phát hiện được. Mỗi cột mốc đều đạt được, sẽ cho bạn thông tin xác định; Nếu cột mốc dự kiến đã đến, nhưng phần mềm lại phải qua *N* tuần mới đạt được tiêu chuẩn chất lượng cần có, bạn cần có lập trường để thúc nhóm cố gắng: "Tôi không biết ngày đưa sản phẩm của chúng ta sẽ bị kéo dài bao lâu, nhưng cột mốc này đã cho chúng ta biết rằng mình tụt hậu *N* tuần".

NGUYÊN TẮC 35: MỌI THÀNH VIÊN ĐỀU ĐẠT TỚI CỘT MỐC KHÔNG HỤT

Nobody reaches the ZD milestone until everybody does

Nếu khó khăn của một nhóm nhiều, làm tốc độ chậm lại, mà các nhóm khác đã hoàn thành công việc của mình, tốt nhất hãy để mọi người giúp đỡ nhóm đang làm chậm này; Công việc trong nhóm chậm đều là hiện tượng không tốt; Công việc của tất cả các nhóm đều được hoàn thành mới coi như xong việc, chỉ cần có một bộ phận chưa hoàn thành tức là đã thất bại.

Có một trường hợp hợp không tốt là (Tôi đã thực sự gặp phải trường hợp này) "Giậu đổ bìm leo": Khi một nhóm làm nhanh thấy nhóm khác làm chậm đang phát triển đang phát triển kỹ thuật then chốt gặp phải khó khăn, có thể làm kéo dài cột mốc, liền yêu cầu nhóm kia đưa thêm càng nhiều tính năng vào. Để tránh trường hợp này, lãnh đạo phải cho mọi thành viên (cả nhóm) có trách nhiệm cùng đạt tới cột mốc, nếu tất cả chưa đạt tới cột mốc thì cũng như chưa có ai đạt tới mốc.

NGUYÊN TẮC 36: SAU MỖI CỘT MỐC, CẦN BÌNH TĨNH KIỂM ĐIỂM

Every milestone deserves a no-blame postmorterm

Sau mỗi lần đạt được cột mốc, cần phải làm ngay một lần kiểm điểm, điều này không cần nhiều thời gian, nhưng lại làm cho nhóm tốt lên. Khi chúng ta nhận thấy đã làm được 100%, có phải thực sự không bỏ sót gì không? Cách kiểm điểm tốt không phải là chỉ trích ai đó kéo dài tiến độ. Phụ trách dự án có thể triệu tập cuộc họp, mỗi nhóm nhỏ tối thiểu có một người tham gia, hoặc bất cứ ai có tâm đắc gì đều có thể gửi email cho phụ trách dự án, kết quả kiểm điểm sẽ được phụ trách dự án gửi email lại cho các thành viên tham khảo. Nhất là những việc làm tốt, cần được nhấn mạnh trong cuộc họp, mọi người sẽ biết sau này phải làm gì cho tốt nhất. Sau mỗi cột mốc làm một lần kiểm điểm, sẽ là cách tốt nhất cho nhóm học tập.

Sau cột mốc chỉ trích ai đó, hoặc nhóm nào đó làm kéo dài tiến độ, là cách kiểm điểm rất ấu trĩ. Mục đích thảo luận tiến độ tụt hậu chỉ nhằm: Nghiên cứu cách phòng tránh đợt sau, tăng cảnh giác của mọi người đối với tụt hậu, cách bổ cứu khi chẳng may tụt hậu. Hậu quả xấu do tiến độ tụt hậu sẽ ảnh hưởng đến tất cả mọi người. Tụt hậu làm tăng giá thành của công ty, các

thành viên không nên nghĩ ngây thơ rằng mình chẳng bị ảnh hưởng gì, do đó quản lý không cần tìm cách trừng phạt, chỉ cần cho mọi người thấy rõ tụt hậu gây tổn hại như thế nào, lần sau phải đặc biệt chú ý.

Chỉ trích ai đó kéo dài tiến độ của mọi người thì thật ấu trĩ, cũng chẳng khác gì chỉ trích chiếc lá không chịu rơi khỏi cây vậy.

Đối với thành viên cá biệt, tiến độ kéo dài là không tốt, nhưng cũng có ý nghĩa giáo dục nhất định. Thông thường, chỉ cần là nhóm lành mạnh, dù tổ chức thành viên cao thấp thế nào hay tình hình kéo dài nghiêm trọng ra sao, thành viên gây kéo dài dây dưa sẽ cảm thấy xấu hổ mà chủ động tìm cách giải quyết, anh ta sẽ trải qua cách kỳ vọng bản thân, cải thiện hiệu quả và phương pháp làm việc. Quản lý phải giúp đỡ anh ta, chỉ trích chỉ là hành vi ấu trĩ, chẳng khác gì chỉ trích chiếc lá không chịu rời khỏi cây vậy.

Nếu sau mỗi cột mốc bạn đều tự kiểm điểm đầy đủ, và các thành viên đều tiếp thu được các bài học qua mỗi cột mốc, nhóm sẽ dần trưởng thành, nhằm tránh lỗi sau này.

NGUYÊN TẮC 37: NẮM VỮNG Ý NGHĨA VÀ TINH THẦN THỰC CHẤT CỦA CỘT MỐC

Stick to both the letter and the spirit of the milestones

Cách làm cột mốc có vẻ khá cao về giá thành, mà nhóm cũng chịu nhiều đau đớn, nhưng đây là phương thức duy nhất đảm bảo thành công. Để phối hợp đưa ra chuẩn tắc đo lường cột mốc, thời gian và sức lực của mỗi thành viên bỏ ra chính là giá thành vượt trội của cột mốc; Để biểu thị cụ thể cột mốc cần phải sửa đổi tính năng, lý tưởng của nhóm có thể bị sụt giảm, trong lòng cũng có đôi chút thất vọng đau đớn. Đau đớn nhất bắt nguồn từ việc tìm cách để mọi người tập trung vào cột mốc, phối hợp với mọi người có giá trị quan thống nhất, chấp nhận bỏ ra nhiều thời gian và sức lực hơn. Cho dù có thực sự tung sản phẩm ra sớm hơn, chúng ta lại phải "huy động binh lực" luyện tập, trải qua sản phẩm trung gian mà ai nấy đều vất vả.

Nhưng lợi ích của cột mốc không hụt sẽ nhiều hơn tất cả những thứ phải bỏ ra, những kinh nghiệm và trưởng thành của mỗi người cũng nhiều hơn đau đớn của họ. Sự thực cột mốc không hụt làm quá trình đánh giá phần mềm được bắt đầu từ giai đoạn bố cục, nhờ đó mà nhóm có nhận thức chung; Hiểu biết về sự thực và trở thành giá trị cao nhất của nhóm. Định nghĩa nội

dung cột mốc, chính là kỳ vọng của mọi người với công việc của mình. Xét về một ý nghĩa nào đó, cột mốc đại diện độ tin cậy với lời hứa của nhóm.

Sau vài cột mốc trong giai đoạn đầu của dự án, trong nhóm đã có cảm giác sứ mệnh đối với cột mốc, có thể phán đoán chính xác mình có thể thực hiện những gì, trong một thời gian hạn chế việc gì có thể hay không thể hoàn thành. Tôi đã trải qua vài cột mốc trên giấy tờ, nhóm coi cột mốc như hình thức và sự giáo điều, mà không thực hiện ý nghĩa và tinh thần thực chất của cột mốc. Theo kinh nghiệm của tôi, nếu là nhóm mạnh, họ sẽ đều có cảm thụ rất mạnh về tính đồng nhất (Xem nguyên tắc 29), có thể tránh được các sự việc lừa dối bản thân hay né tránh hiện thực, tự hy vọng mọi việc đều là sự thật chứ không bị ép tuân thủ bất cứ một quy định hay giáo điều truyền thông nào. Nhóm mạnh sẽ hy vọng cột mốc và các thành viên có tính đồng nhất, không hình thức hoá cột mốc, hay nghiêm khắc quá mà không thực hiện được. Nói thực đây là phẩm chất đạo đức quan trọng của nhóm phát triển thành công, tôi đã từng thấy các thành viên thách thức lẫn nhau xem mình có thực sự tự lừa dối bản thân mà không làm được việc hay không; Trong nhóm mạnh, thành viên có thể "ngửi" được đối phương có phóng đại hay không.

Tôi đã thấy các thành viên thách thức lẫn nhau, có phải đã đánh giá quá cao bản thân trong cột mốc mục tiêu giai đoạn đầu.

Trong cột mốc giai đoạn đầu của dự án họ yêu cầu xuống thấp một chút thì có thể chấp nhận được, tính năng khó thì không đạt yêu cầu hoàn thành trong cột mốc này, tiêu chuẩn chất lượng không đặt cao quá, tài nguyên có thể nhiều hơn một chút, những điều này đều không cấp thiết lắm, có thể coi như chi phí đánh đổi của nhóm. Ngược lại nếu coi nhẹ các vấn đề đã phát sinh mà không bổ cứu, hoặc tự dối lừa về các vấn đề phát triển của nhóm thì sẽ gây sai lầm lớn. Chỉ có không ngừng kiểm nghiệm bản thân trong giai đoạn đầu của dự án, nhóm mới có thể phát triển, đồng thời nắm vững bí quyết thành công trong các cột mốc tiếp theo.

Đã có lần tôi tham gia vào một nhóm, do đặt thời gian biểu quá tự tin, mục tiêu công việc cũng quá cao nên gần như phải hoàn thành trong đoạn đầu, kết quả là đã kéo dài một cột mốc (Dưới đây gọi tắt là M1), lúc này ai cũng căng thẳng, cũng may mà nhóm cũng thuộc loại mạnh, chúng tôi cũng biết thành công trước mắt không có nghĩa là thành công sau này, chúng tôi tự nguyện khiêm tốn kiểm điểm.

Chẩn đoán xem có rắc rối không.

Trong giai đoạn M1, nhóm đã chịu đựng sự đau khổ mục tiêu quá lớn. Công việc phát triển phần mềm trong giai đoạn M1 làm chúng tôi phát hiện ra một số tính năng mẫu mực (Tham khảo nguyên tắc 3) có chỗ không được thiết kế chi tiết, không hỗ trợ đầy đủ thông tin cần truyền đạt, thực tế thiết kế như vậy không thể đạt được mục đích thay đổi thói quen người sử dụng. Thế là chúng tôi tranh luận kịch liệt, gần như làm chia cắt nhóm. Cuối cùng chúng tôi cũng có kết luận, phương án thiết kế quá mới có thể làm cho mọi người phấn chấn, nhưng cũng tăng gánh nặng công việc của toàn nhóm lên rõ rệt.

Vốn là mục tiêu được sửa đổi tốt hơn, hành động có ích hơn cho phần mềm nhưng lại gây ra tác dụng phụ: Chỉ còn ba tuần nữa là đến M1, thành viên vẫn không thể xác định là có cần phát triển tính năng mới

thiết kế không? Trong thiết kế cũ có phải cần loại bỏ vài tính năng không? Các thành viên không làm rõ được cấp độ ưu tiên của các tính năng này, nếu dùng phương án thiết kế mới thì ai sẽ làm và làm vào lúc nào. Thiết kế cũ có lẽ không lý tưởng, nhưng đó là nhận thức chung mà khó khăn lắm chúng tôi mới có được, chẳng lẽ như vậy lại phá bỏ lập lại mục tiêu mới? Trong trường hợp này quản lý có duy trì nguyên tắc trao quyền không (Nhất là khi các tính năng này vốn là chủ trương của họ), hơn nữa không thấy rằng chúng tôi đang bị phiền nhiễu? Các vấn đề không xác định như vậy nhiều đến nỗi không liệt kê nổi. Rất dễ thấy, những vấn đề này làm chúng tôi ngày càng xa rời mục tiêu M1.

Cứ coi như lượng công việc không tăng, nhưng việc nâng cao mục tiêu như vậy sẽ gây ra tranh luận kịch liệt. Khi chúng tôi thảo luận phương án thiết kế mới, đều không nghĩ rằng nó đem đến nhiều vấn đề như vậy, nên đã bất giác tăng thêm rất nhiều tính năng, chẳng khác gì phá vỡ giao ước của nhóm. Đúng vậy, xét về kỷ luật nhóm thì chúng tôi rất dở, nhưng chúng tôi cũng phân tích triệt để từng tính năng trong thiết kế mới, nghiên cứu ảnh hưởng của nó đối với từng người hay từng sự việc, và có được nhận thức chung mới, mọi người đều đồng tình các tính năng thêm vào đều rất quan trọng, nếu không có thiết kế này thì sản phẩm sẽ thiếu sót. Chúng tôi kết luận sẽ hy sinh các tính năng

thứ yếu khác, sắp xếp lại nhân lực, cố gắng giảm thiểu những ảnh hưởng bất lợi đến tiến độ.

Bây giờ, chúng tôi đối diện với kết quả của phương án: Sau khi tranh luận và quyết định sử dụng thiết kế mới, chúng tôi trở lại hiện thực của kế hoạch phát triển, chúng tôi đã đặt sai M1, hơn nữa thời gian cũng rất đáng sợ, chúng tôi họp quản trị dự án và nhân viên bảo đảm chất lượng của cả công ty, đánh giá triệt để về hiện trạng. Kết quả của lần thảo luận này, cho thấy mọi sự việc trong phát triển phần mềm đều không thể biết trước được, nhưng vẫn có thể quản lý thích đáng - tôi rất thích điểm này.

Tranh luận không ngừng

Có một số người cho rằng, thực ra chúng tôi có thể hơi nới lỏng một số kỳ vọng của M1, theo thiết kế cũ đạt đến mục M1, trên cơ sở không ảnh hưởng mục tiêu đã định, bớt làm vài tính năng. Dẫu sao đây cũng là cột mốc đầu, nếu không đạt được về ý nghĩa thực chất thì tối thiểu cũng làm được về mặt hình thức.

Ý kiến thứ hai cho rằng, rất khó biết được có đạt đến M1 không, mới bắt đầu đã đặt mục tiêu cao như thế, vậy thì ý nghĩa mục tiêu M1 cũng thể giải thích theo nhiều kiểu khác nhau. Tuy về cơ bản chúng tôi đồng ý cần đạt được M1, nhưng thực chất lại dùng thiết kế mới và phá vỡ giao ước.

Ý kiến thứ ba cho rằng, do chúng tôi không làm kịp thời gian M1 nên làm bừa, họ cho rằng đó là do chúng tôi cho phép xuất hiện quá nhiều chức năng thừa trong mục tiêu M1, bây giờ lại muốn che lấp đi. Chúng tôi đã lẫn lộn, mơ hồ, lừa dối bản thân, tự cho rằng nhóm có kỷ luật, tự cho rằng trong ngoài đồng nhất, nhưng thực tế chúng tôi thiếu khả năng kiểm soát tính năng mà không dám thừa nhận sự thực không thể đạt được M1.

Sau cùng, có người chỉ ra rằng, chúng tôi không hề bị rắc rối (Seaturistis), rắc rối là khi cho thêm nhiều tính năng không có mục tiêu rõ, hơn nữa lại là tính năng nhỏ, đồng thời không liên quan đến mục tiêu sản phẩm và nhận thức chung. Còn các tính năng có thêm trong thiết kế mới có chúng tôi lại là nên có trong sản phẩm, chỉ đáng tiếc là các tính năng này quá nhỏ, dẫn đến bị bỏ qua trong thiết kế cũ. Nhưng nếu ghép các tính năng này lại sẽ có ảnh hưởng rất tốt cho sản phẩm. Mà hiện này phát hiện được trong giai đoạn phát triển M1 nên phải tuyên bố sự kéo dài M1. Còn nếu không phát hiện ra vấn đề này thì sản phẩm vẫn bị trì hoãn.

Cũng như bệnh tật rất dễ xâm nhập cơ thể khoẻ mạnh, rắc rối trên có thể gây thương tổn cho nhóm chưa đủ mạnh - có lẽ sẽ trở thành lạc hậu trên thị trường, hoặc là nhóm không hiểu rõ về bản thân. Trong nhóm

mạnh, khi tiến độ lạc hậu sẽ có cảm giác bất an nhưng chỉ có nhiệt huyết và niềm tin lớn mới có thể tin vào tầm quan trọng của các tính năng mới, và sau đó sẽ khắc phục được khó khăn. Mặt khác, tính bất định và gánh nặng công việc quá nhiều, sẽ làm suy giảm hệ thống miễn dịch của nhóm, làm giao động niềm tin, mà không dám khẳng định rốt cuộc những tính năng mới thêm này có phải dù trọng tâm không thể thiếu của sản phẩm. Nếu lúc này thị trường hay khách hàng cũng thay đổi, làm cho nhóm tăng thêm nhiều tính năng hoặc định nghĩa lại sản phẩm, cuối cùng sẽ dẫn đến thất bại.

Các tính năng mới quyết định trong M1, đều được phân tích triệt để, mọi người đều có nhận thức chung, nguyện tăng gánh nặng công việc cho lý tưởng này, tất cả các đồng nghiệp quan trọng đều đồng ý về các tính năng này nên cần phải kéo dài tiến độ. Nhưng nếu không có tính năng mới, chúng tôi sẽ không thể thay đổi thói quen sử dụng của khách hàng, cũng mất đi ý nghĩa của tính năng mẫu mực. Do đó, phải điều chỉnh tiến độ và nhân viên, nhưng ai cũng đồng tình, phát triển các tính năng là quyết định đúng đắn.

Bản chất M1

Kết luận trên có ý nghĩa đặc biệt với M1, chúng tôi gọi kinh nghiệm này là bản chất M1, để phân biệt với

hành vi nhóm kiểu bệnh thái. Chúng tôi không cho rằng đó là rắc rối, vì chúng tôi đã phân tích các tính năng mới một cách xác thực, triệt để và không tiếc công sức, đây chính là khác biệt lớn nhất. Ngoài ra, phương án thiết kế này, thực sự có sửa đổi quan trọng đối với những khuyết điểm của thiết kế cũ, mà những đóng góp này còn hơn hẳn ảnh hưởng phụ cho kỷ luật kém.

Về cơ bản chúng tôi đã hiểu, M1 là thời cơ loại bỏ những vấn đề này, M1 cho chúng tôi phát hiện những chỗ mà khi thiết kế còn mơ hồ hay bỏ qua, nhược điểm của bảng tiến độ, và tái tập trung nhận thức mọi người, có khái niệm rõ ràng hơn về mục tiêu. Từ kinh nghiệm này, tôi kết luận: Từ nay về sau mọi người đều rất chú ý M1, sửa mục tiêu M1 cần phải có nhận thức chung của nhóm, và thực sự có thể đạt được.

NGUYÊN TẮC 39: QUÁ NHIỀU CỘT MỐC SẼ KHÔNG NẮM VỮNG

A handful of milestones is a handful

Nhiều người cho rằng, 6 tuần đến ba tháng là khoảng cách thời gian lý tưởng giữa các cột mốc. Nếu nhóm của bạn nhỏ (khoảng 10 đến 20 người) hay mục tiêu ít có thể dùng khá nhiều cột mốc vì gánh nặng vượt trội của cột mốc đối với nhóm nhỏ sẽ rất ít. Mà nếu nhóm nhỏ trải qua nhiều cột mốc, sẽ trưởng thành nhanh chóng và tập trung được văn hoá chung (Tham khảo nguyên tắc 7), đây là những lợi ích kèm theo.

Nhưng theo cách nhìn nhận của tôi, số cột mốc của bất cứ nhóm nào vượt quá 7 đều là quá nhiều, tối thiểu cũng là hiện tượng bất thường. Vì thông thường, không ai có thể dự đoán được tương lai xa như vậy. Kinh nghiệm của tôi là 3 đến 4 cột mốc, với một cột mốc tung sản phẩm ra, là con số lý tưởng nhất (Tham khảo nguyên tắc 18).

NGUYÊN TẮC 40: MỖI CỘT MỐC CẦN CÓ TÔN CHỈ RIÊNG

Every little milestone has a meaning (story) all its own

Mỗi cột mốc, dù to hay nhỏ, cần có lý do thành lập nó. Vẫn còn nhớ khi thảo luận về bản chất M1, đề cập đến những tranh luận kịch liệt trong nội bộ nhóm, chúng tôi mong muốn đạt được tinh thần thực chất M1. Tôn chỉ của cột mốc là xây dựng lý do cho nó, có thể rút gọn biểu đạt thành một hay hai câu, đồng thời mô tả ý nghĩa đạt được cột mốc đối với nhóm. Xét về lý luận, bất cứ tranh luận nào về cột mốc có thể quay về tôn chỉ của nó để tìm được giải đáp. Tôn chỉ của cột mốc có thể mô tả rõ ràng mục tiêu của nó, tôi cố ý không gọi là mục tiêu của cột mốc vì tôn chỉ của cột mốc nên bao gồm đồng thái giữa nhóm và nó, tinh thần nhóm, và các thông tin ẩn chứa của cột mốc, cũng chính là bao gồm ý nghĩa toàn diện của nó. Nếu chỉ xét mục tiêu M1, đa số mọi người đều không có động cơ đạt được nó, hay nghĩ đến thù lao của nó, mục tiêu M1 là hàng dãy thực đơn, là những công việc mà chúng tôi phải làm (Tuy mục tiêu cũng có thể sai, phần mềm của ai lại có thể không xác định chứ). Một tờ hóa đơn khó có thể làm mọi người nhiệt tình hay chú ý, để có thể làm ra phần mềm lý tưởng. Mọi người chỉ nguyện hy sinh cống hiến cho một

sự việc có ý nghĩa, chỉ có tôn chỉ của cột mốc mới có thể làm họ nguyện dốc sức lực, tâm trí của mình để đổi được kết quả nào đó.

Trong quá trình phát triển Visual C++, có tài tôn chỉ cột mốc khá tốt, đáng để tham khảo. Trong cột mốc điển hình đầu tiên, chúng tôi thường đặt mục tiêu trên "*lý luận thức ăn cho chó*" (dog fooding), ý nghĩa là phải sử dụng sản phẩm phát triển được mới biết nên phát triển tiếp như thế nào. Việc này bắt nguồn từ lý luận tiêu thụ của Teve Ballmen, được Microsoft phát triển, câu danh ngôn của ông là: "Xưởng thức ăn cho chó cần thử ăn thức ăn do mình sản xuất. Chỉ có tự trực tiếp thử sản phẩm của mình, mới có thể hiểu thực sự về sản phẩm của mình làm ra". Đối với công cụ lập trình như Visual C++, khái niệm này rất quan trọng, vì nó được viết ra bằng chính công cụ lập trình của nó trong thời kỳ đầu. Đối với nhóm của chúng tôi, ý nghĩa của việc triệt để thực hiện lý luận thức ăn cho chó là phiên bản mới nhất định phải có sức sản xuất hơn phiên bản cũ, cho dù phiên bản mới vẫn chưa hoàn thiện hoặc còn lỗi. Trông có vẻ chúng tôi không nhấn mạnh lý luận thức ăn cho chó này: Người quản lý không phải ngày nào cũng đi kiểm tra văn phòng xem mọi người có dùng thử phần mềm do mình phát triển hay không. Nhưng toàn thể thành viên nhóm đều nhận thức rõ tầm quan trọng của lý luận đó: Nếu mỗi chương trình kết hợp đều thực

hiện được tinh thần lý luận thức ăn cho chó, sản phẩm của chúng tôi sẽ ngày một tốt hơn. Sau đó, thường xuyên sử dụng phần mềm do mình phát triển, chúng tôi đã tìm ra tất cả lỗi và tỳ vết, sức sản xuất được nâng cao, và cũng hiểu ra việc nắm vững hiệu suất thiết kế.

Người thử dùng phiên bản mới thường được lựa chọn kỹ.

Ngoài ra khi xây dựng tôn chỉ cho cột mốc chúng tôi đặc biệt chú ý "lúc nào giao cái gì cho ai". Để bảo vệ danh dự cho mình, khi giao phiên bản ra đều rất cẩn thận, thông thường đối tượng đều được lựa chọn, vì giao nhầm thời cơ cho nhầm người đều làm tăng gánh nặng vượt trội cho nhóm. Do đó, có thể nói cột mốc chính là chúng tôi chuẩn bị sản phẩm trung gian trao cho một số khách hàng thời kỳ đầu thử sử dụng .

Do đó, tôn chỉ cột mốc đầu tiên của chúng tôi là: "**Tất cả các chương trình kết hợp đều được nhóm phát triển thử dùng, đồng thời chuẩn bị sản phẩm trung gian giao cho đơn vị trong Microsoft dùng thử**". Cách nói đơn giản là "thử ăn thức ăn cho chó". Cuối cùng, tôn chỉ cột mốc (Cột mốc thứ 3) có lẽ là: "Các tính năng đều đã hoàn thành, chuẩn bị chuyển cho một số khách hàng bên ngoài dùng thử".

Điểm mấu chốt của tôn chỉ cột mốc là:

- Tôn chỉ phải chỉ ra tinh thần của cột mốc.

- Phán đoán cột mốc đó có đạt được không, xem tôn chỉ cột mốc có được thực hiện hay không.

- Trước khi bắt đầu công việc, mọi người đều hiểu rõ và chấp nhận tôn chỉ của cột mốc đó.

NGUYÊN TẮC 41: TÌM KIẾM CỘT MỐC XUẤT HIỆN TỰ NHIÊN

Look for the natural milestones

Tuy rất khó có thể ghi lại quá trình phát triển phần mềm một cách chính xác theo thứ tự thời gian, nhưng có một số sự việc chắc chắn sẽ phát sinh, có tác dụng phân chia hoạt động phát triển, được coi như cột mốc xuất hiện tự nhiên. Hãy chú ý cái tôi gọi là cột mốc xuất hiện tự nhiên, là chỉ "bình thường" chứ không phải "hay gặp", bình thường có ý nghĩa là bản thân quá trình phát triển không có vấn đề lớn. Dưới đây là 6 loại cột mốc xuất hiện tự nhiên:

1. Thiết kế sản phẩm có xu hướng ổn định.

2. Sản phẩm trung gian được định nghĩa rõ ràng.

3. Nhóm thực sự hiểu cần tốn bao nhiêu thời gian sức lực mới hoàn thành được mục tiêu (Thường phát sinh nhiều lần, nhất là khi tiến độ tụt hậu).

4. Thiết kế sản phẩm bị cắt giảm, hoặc tăng tài nguyên, hoặc lỡ tiến độ, hoặc cả ba cùng xảy ra.

5. Ngừng hoạt động phát triển.

6. Sản phẩm bước vào giai đoạn loại lỗi hay ổn định.

Sáu cột mốc tự nhiên này không thể chỉ xuất hiện trong cả quá trình phát triển, mà cũng có thể xuất hiện trong một cột mốc. Phía sau chúng ta sẽ thảo luận chi tiết, nhưng mọi người hãy chú ý, hành động của mỗi cột mốc cần tương đồng với hành động của tất cả các cột mốc cộng lại, cũng như với hành động của cả kế hoạch kỹ thuật.

Hành vi biểu hiện trong từng cột mốc của một nhóm mạnh cần được dự kiến, gọi là mô hình tự nhiên của cột mốc, người quản lý cần coi trọng và bồi dưỡng. Nếu mô hình tự nhiên của cột mốc không xuất hiện, chứng tỏ nhóm có vấn đề, người quản lý phải có hành động chữa trị. Cho đến nay, các cột mốc tự nhiên mà tôi quan sát được có thể quy nạp thành 6 loại, nhưng các sự việc có thể duy trì lâu dài khó có thể dự đoán được. Điều có thể xác định được là, sự việc trước kết thúc, sự việc sau phải xuất hiện ngay, giữa các sự việc trước sau có thể lặp lại hoặc hầu như cũng xảy ra; Những sự việc này có thể lần lượt xuất hiện hoặc cùng xuất hiện, tuỳ vào quan hệ hoà hợp của nhóm, nhóm hoà hợp mạnh, thông tin sẽ truyền tải nhanh chóng, 6 sự việc rất dễ cùng xảy ra.

CỘT MỐC THỨ NHẤT: THIẾT KẾ CÓ XU HƯỚNG ỔN ĐỊNH

Khi mới bắt đầu, thiết kế sản phẩm gồm nhiều tính năng là rất trừu tượng, là thứ làm người ta vừa hưng phấn vừa mê hoặc, hưng phấn về cách nghĩ hay sáng tạo của bản thân được thực hiện, mê hoặc vì không xác định được cuối cùng nó thuộc mô thức nào. Mọi người (kế hoạch kỹ thuật, nhận thức chung đã được hình thành trước cột mốc) đều biểu thị các tính năng này cần được hoàn thành, nhưng khi thực sự có vấn đề, lại không ai biết ai, làm gì và khi nào mới hoàn thành các tính năng này. Tình trạng này được tôi gọi là "Triệu chứng giấc mơ phần mềm" (The software dream syndrome). Trong giai đoạn thiết kế sản phẩm, hiện tượng chi tiết không rõ này sẽ không ngừng xuất hiện, cho đến khi hình thành phân công công việc chi tiết mới thôi.

Có thể bạn nghe thấy nhân viên phát triển nói: "Đúng vậy, đương nhiên chúng ta cần đưa chức năng *footbar* vào cột mốc đầu tiên; Tôi đang làm mô hình sơ lược cho anh xem".

Còn nhân viên bảo đảm chất lượng có thể nói: "Tính năng *Widget* cần bắt đầu ở cột mốc 2, nhưng tôi không biết để ai phát triển".

Theo truyền thống, hoặc có thể nói tối thiểu trong công ty có chế độ, đều có mấy tập tài liệu về phát triển chương trình, bao gồm định nghĩa quy cách kho dữ liệu, sơ đồ lôgic của chương trình, ...nhiều không kể hết. Tuy các tài liệu này có tác dụng cụ thể hoá các sự việc cần làm của chương trình, làm rõ các chi tiết, nhưng do các lí do sau, tôi phản đối mất thời gian làm các tài liệu sau:

+ Trước khi thiết kế có xu hướng ổn định phải lần lượt sửa đổi, làm cho tài liệu được làm lại, kết quả cập nhật tài liệu trở thành một loại đạo đức hoặc phụ tùng mệnh lệnh chứ không phải nhu cầu thực tế.

+ Tác dụng của tài liệu là xác định mọi sự việc trước khi bắt đầu công việc phát triển, kết quả cũng có thể hạn chế việc phát triển chương trình. Rất nhiều sự việc liên quan đến phần mềm đều ở trong quá trình phát triển mới biết được nên làm như thế nào, hơn nữa lúc này mới biết làm thế nào là tốt nhất, đôi khi nhân viên phát triển có những sáng tạo bất ngờ, làm sản phẩm thêm đặc sắc. Do đó, nếu loại bỏ các bộ phận chưa biết trong phần mềm, chẳng khác gì tuyên bố phần mềm chấm hết, không có sự bắt đầu.

Trong quá trình phát triển phần mềm, việc quan trọng nhất không phải làm những gì bạn đã giao ước, mà trong điều kiện thời gian và tài nguyên có hạn, đưa

ra lựa chọn thông minh nhất trong các phương thức có khả năng để có được kết quả tốt nhất.

Thiết kế sản phẩm sẽ dựa vào các hình thức như email, định nghĩa quy cách, mô hình sản phẩm, họp nhóm để thảo luận, làm cho biến động có xu thế ổn định, khi thiết kế sản phẩm dần được xác định, giai đoạn tiếp theo có thể chuẩn bị bắt đầu. Quản lý cần chú ý xem các tranh luận đã tan biến do nhận thức chung chưa, kiểm tra xem các tính năng đã có trong thiết kế chưa, nhận thức của mọi người đối với thiết kế sản phẩm có giống nhau, nếu đáp án đều là "Đúng" thì có thể bước vào giai đoạn sau, nếu không thiết kế sản phẩm chưa được coi là chín muồi.

CỘT MỐC THỨ HAI: SẢN PHẨM TRUNG GIAN ĐƯỢC ĐỊNH NGHĨA RÕ RÀNG

Nếu có thể dùng phương thức rõ ràng, đơn giản và đáng tin để biểu đạt sản phẩm trung gian mà không có gì mơ hồ, phức tạp, thì có thể nói "Sản phẩm trung gian được định nghĩa rõ ràng". Khi dự án chính thức bước vào giai đoạn bắt đầu, hoặc khi cần kiểm tra xem có đạt được cột mốc không, chúng ta cần phải có định nghĩa rõ ràng: sản phẩm cần có mô hình như thế nào, cần hoàn thành công việc gì, do ai phụ trách, yêu cầu chất lượng đến mức độ nào, tất cả đều phải có giá trị ước lượng rõ

ràng, tuy không phải rõ như ảnh chụp nhưng tối thiểu rõ như tranh vẽ của phái ấn tượng.

Cùng với việc xác lập thiết kế sản phẩm, nhân viên bảo đảm chất lượng, phụ trách dự án, nhân viên phát triển và nhân viên tài liệu có thể hợp thành một nhóm nhỏ, từng nhóm nhỏ sẽ đảm nhiệm phát triển cụ thể hóa, mô tả rõ hơn nhu cầu và mục đích của họ. Họ bắt đầu bỏ nhiều thời gian thảo luận và hoà hợp bởi họ có mục tiêu và công việc chung, họ cùng quan tâm một sự việc, nhận thức về thiết kế sản phẩm cũng tương đồng, thế là các công việc chi tiết dần dần tự hình thành trong lòng thành viên nhằm sắp xếp công việc theo trình tự thời gian, vẽ ra bản nháp tiến độ.

CỘT MỐC THỨ BA: NHÓM THỰC SỰ HIỂU CẦN TỐN BAO NHIÊU THỜI GIAN VÀ SỨC LỰC MỚI HOÀN THÀNH ĐƯỢC MỤC TIÊU

Đương nhiên, trước khi hoàn thành công việc phát triển, bạn không thể biết được thời gian cần thiết. Khi thiết kế sản phẩm được định, sản phẩm trung gian được định nghĩa rõ, thành viên sẽ có một cảm giác lạc lõng: Thứ nhất, bề ngoài sản phẩm không đẹp như tưởng tượng; Thứ hai, việc cần làm thực tế nhiều hơn so với tưởng tượng; Thứ ba, nhóm phát hiện ra mình cần nhiều tài nguyên hơn, ít mục tiêu hơn và nhiều thời gian hơn.

Lúc này, dù nhóm có kinh nghiệm phong phú như thế nào đi nữa, lãnh đạo có xuất chúng thế nào đi nữa thì bầu không khí thất vọng vẫn phảng phất, đó là một cảm nhận u uất và sa sút, gần như mình bị hạ gục vậy. Thực ra, đó là hiện tượng tốt, biểu thị thành viên có nhận thức hơn về sự thật, sau này sẽ cảm thấy vô tư vì hiểu rõ sự thực. Ảo mộng là bắt đầu của trưởng thành, "Hội chứng giấc mơ phần mềm" tan biến. Trong một nhóm mạnh, nhận thức về sự thực sẽ dẫn đến một cao trào hành động mới, nhóm của chúng tôi gọi đó là "hội nghị chiến tranh" (War room meeting), chúng tôi liên tục mở các cuộc họp kéo dài vài tiếng nhằm kiểm tra tình hình giai đoạn hiện tại và nghiên cứu đối sách khi có vấn đề.

Trong nhóm có một bầu không khí thất vọng, đó là cảm nhận u uất và sa sút, gần như mình bị hạ gục vậy.

Trong ba cột mốc tự nhiên trước, sự việc quan trọng nhất là: giải quyết rõ ràng những thứ chưa biết. Vậy mà, khi sự việc sáng tỏ, công việc thường chỉ có nhiều chứ không ít (Bộ mặt mơ hồ thường khá đẹp), thành viên sợ hãi vì các thông tin có được, nhưng nhóm mạnh sẽ đối mặt được với tình hình này, tìm cách khắc phục sợ hãi.

CỘT MỐC THỨ TƯ: THIẾT KẾ SẢN PHẨM BỊ CẮT GIẢM, HOẶC TĂNG TÀI NGUYÊN, HOẶC LỠ TIẾN ĐỘ HOẶC CẢ BA CÙNG XẢY RA

Trong giai đoạn này, nhóm sẽ lợi dụng "Tam giác phần mềm" (Tham khảo nguyên tắc 28) để giải quyết xung đột gặp phải. Nếu mọi chuyện thuận lợi, lúc này vừa vặn là một trong bốn phần đầu của cả dự án (hoặc cột mốc của ai đó). Chúng ta có lý do để tin rằng, càng là nhóm mạnh, ba cột mốc đầu đi càng nhanh, đồng thời có nhiều thời gian giải quyết xung đột mà tam giác phần mềm cho họ biết, hoặc xây dựng lại nhận thức chung, đây cũng là lúc phải sửa đổi một chút kế hoạch cột mốc. Nếu mọi chuyện thuận lợi, tuy có thể phải điều chỉnh kỳ vọng nào đó về cột mốc - có lẽ phải bỏ vài tính năng, hay tăng thêm chút tài nguyên, ... tôn chỉ cột mốc cần duy trì bất biến.

Trước khi đạt đến cột mốc, không ai đảm bảo bạn đạt đến đúng hạn.

Trong lúc này một nhóm mạnh sẽ thể hiện sức bật và khả năng phản ứng, để có thể đạt được yêu cầu của cột mốc này. Bạn không muốn cột mốc bị kéo dài, đó là biện pháp cuối cùng; bạn cũng không muốn tài nguyên bị yêu cầu tăng vô hạn. Lúc này, điều duy nhất bạn muốn thấy là hàng đống giao ước hình thành giữa các thành viên, bồi dưỡng được hiệu suất giải quyết vấn đề,

chuẩn bị tốt để biểu hiện mô hình hành vi cột mốc xuất sắc nhất.

CỘT MỐC THỨ NĂM: NGỪNG HOẠT ĐỘNG PHÁT TRIỂN

Vào một điểm thời gian nào đó, các hoạt động viết chương trình đều dừng lại, cũng là lúc kết thúc phát triển các tính năng, lúc này phải ngăn cấm mọi sửa đổi để phần mềm bước vào giai đoạn hoàn thành và ổn định. Sau khi hoàn thành công việc phát triển sẽ không cần viết hay sửa chương trình nữa, nghe có vẻ hơi thừa lời, nhưng cần phải đặc biệt chỉ ra cột mốc này, vì nhu cầu thời gian dự kiến có thể không đúng, cột mốc này tương đương căn cứ để tính thời gian tiêu hao. Cần phải nhớ rằng phần mềm không cần hoạt động phát triển nào nữa mới coi như đạt đến cột mốc này.

CỘT MỐC THỨ SÁU: SẢN PHẨM BƯỚC VÀO GIAI ĐOẠN LOẠI LỖI HAY ỔN ĐỊNH

Nếu bạn có thể tính ra "khả năng sửa lỗi tịnh" (Net bag fix capacity), tức là "tỷ lệ phát hiện lỗi", mà giá trị này là dương thì bạn có thể tính được cột mốc thứ 6 cần bao nhiêu thời gian. Hơn nữa, trên kinh

nghiệm lần này, đối với các dự án tương tự sau này, có thể giảm tỷ lệ phát hiện lỗi, và có tính chính xác cao hơn cho phần mềm. Lúc này bạn phải tìm được giải đáp cho các vấn đề sau: Có bao nhiêu lỗi, bao lâu mới xoá hết lỗi, bao nhiêu mã chương trình phải tái kiểm nghiệm.

Khi thảo luận chi tiết các vấn đề này thì cũng là lúc bạn đã tiếp cận đoạn cuối cột mốc, các lỗi phát hiện được đều bị xoá. Bạn sẽ biết cách sử dụng phương pháp đánh giá đẳng cấp lỗi (Bug triage) để giải quyết vấn đề, đơn giản hơn phương thức thấy là sửa.

NGUYÊN TẮC 42: NẾU TRƯỢT, HÃY ĐỨNG LÊN

When you slip, don't fall

Tiến độ tụt hậu? Đừng có lo, chưa phải ngày tận cùng của thế giới, cũng như cảm mạo vậy, làm người ta khó chịu, nhưng lại biểu thị cơ thể đang tự chữa bệnh.

Xét từ góc độ khác, tiến độ tụt hậu làm nhóm tỉnh lại từ trong ảo mộng, bắt đầu đối mặt với hiện thực. Cũng như một loại tỉnh ngộ, một tâm tư "Bây giờ thì tôi đã hiểu". Khi từ trong giấc mơ tỉnh lại, bạn sẽ nghĩ: "Được, tôi đã tỉnh, vừa rồi tôi đã như thế nào? Ba tháng qua mê muội đến nỗi không phát hiện ra sai lầm rõ ràng này - kéo dài thanh trạng thái? Cuối cùng thì tôi cũng tỉnh".

Nếu coi tiến độ tụt hậu kéo là vấn đề đạo đức, mọi người sẽ tìm mọi cách tránh đối mặt.

Ba tháng sau, có lẽ là ba tuần sau, bạn lại phát hiện ra sự kéo dài rõ rệt, bạn lại nghĩ: "Lần này tôi đã tỉnh rồi. Bây giờ thì tôi đã hiểu". Mỗi lần tỉnh giấc đều làm bạn bước vào thế giới "thực", nhưng có ngày bạn

tỉnh lại từ thế giới "thực" này, sẽ phát hiện "Thực" vừa rồi vẫn chưa thực hoàn toàn.

Bạn có thể coi không ngừng tỉnh giấc như quá trình trưởng thành. Đương nhiên, mỗi lần tỉnh giấc đều cho bạn nhiều kiến thức và kinh nghiệm hơn, giúp bạn càng tiếp cận thực tế, làm phong phú nhân sinh của bạn. Bộ đếm tỉnh ngộ này bắt đầu từ D của dự án mới, mỗi lần tỉnh giấc thêm 1, nếu bạn có được *N* kinh nghiệm tỉnh giấc, bạn sẽ bắt đầu bộ đếm tỉnh ngộ từ *N*.

Bất cứ dự án nào đều gặp phải tiến độ tụt hậu hay tỉnh giấc, nhưng một số tỉnh giấc không đi kèm tiến độ tụt hậu, điều này có thể có nhưng hơi nguy hiểm và cũng không phải là chuyện tốt. Dưới đây là những quan niệm đúng đắn khi tiến độ lạc hậu:

- Tiến độ tụt hậu không liên quan đến đạo đức, hãy ghi nhớ! Đây không phải là thất bại hay làm sai, đáng bị phạt, đây là điều không thể tránh khỏi trong quá trình sáng tạo tài sản trí tuệ. Tuyệt đối không nên có cảm giác phạm lỗi đạo đức hay trách nhiệm lương tâm, và cũng yêu cầu nhóm của bạn nghĩ như vậy. Bởi trách nhiệm sợ nhất là bị trách cứ, họ có thể do đó mà bị áp lực tinh thần, khi thảo luận thường không đi đúng chủ đề, thấp thỏm lo âu giữa phạm lỗi và khát vọng vinh quang, kết quả là muốn cố hết sức loại bỏ áp lực này, suy nghĩ lung tung, vì vậy mà tiêu hao hết sức

lực. Do đó, tuyệt đối không để ai đó liên tưởng vấn đề đạo đức đối với tiến độ tụt hậu, cho rằng đây là việc không nên hoặc không đúng.

- Không được che giấu sự thật. Khi gặp xung đột hay nguy hiểm, ai cũng mong rằng không có chuyện này và có xu hướng né tránh khi tiến độ tụt hậu, bạn và thành viên đều không muốn nói về nó, nhưng lúc này cần khắc phục thiên tính này, phát huy khả năng lãnh đạo siêu việt của bạn, thay đổi sự việc. Không nên trốn vào phòng của bạn mà lẩm bẩm: "Tôi không dám xem, tiến độ đã kéo dài, ...không, tôi không nghĩ lại có chuyện như vậy... ". Hãy dũng cảm bước ra đối mặt với vấn đề, hãy dẹp lui vấn đề chứ không phải lảng tránh nó.

- Biến lực cản thành hậu thuẫn, lợi dụng tiến độ tụt hậu để khởi phát hiệu suất. Nhận thức về tiến độ tụt hậu là một loại tỉnh ngộ, thành viên lại phấn chấn tinh thần, có thông tin mới nhất, nhìn nhận mới nhất, sáng tạo và khả năng tưởng tượng cũng có thể thay đổi, lãnh đạo phải biết đưa luồng sinh khí mới đến nơi hữu hiệu nhất, không phải băn khoăn về những trói buộc truyền thống, tiến độ tụt hậu là cơ hội tốt nhất để kiểm tra sức bật của nhóm.

Tiến độ tụt hậu không phải là vấn đề, sợ hãi tiến độ tụt hậu mới là vấn đề. Tiến độ tụt hậu không có

nghĩa là độ khó của sản phẩm quá lớn không thể phát triển. Nhưng nếu tiến độ đã bị tụt hậu mà không hay biết, có nghĩa là thành viên không suy nghĩ, không quan sát, không thảo luận, có thể nói lúc này tổ chức đã đứng bên bờ vực.

Biết cách dùng sự kéo dài, sẽ là lúc nhận ra năng lực lãnh đạo của bạn, đây là lúc thành viên yếu đuối nhất và cần bạn nhất, lúc này họ rất dễ tiếp nhận lời bạn nói, khả năng học của họ cũng mạnh nhất. Nếu bạn ngồi trong phòng quát tháo, bạn sẽ mất đi cơ hội thu phục lòng thành viên. Bạn cần phải nói: "Tiến độ tụt hậu rồi, chúng ta hãy xem xem vấn đề xuất phát từ đâu?... Chiều nay 5 giờ tại phòng họp chúng ta sẽ kiểm điểm từng chi tiết, nhất định phải giải quyết vấn đề. Khi thành viên hiểu rằng bạn không muốn tìm cách giải quyết vấn đề, họ sẽ vui vẻ thảo luận, mọi người cùng nhau nghiên cứu vấn đề, tìm cách khắc phục trên mọi góc độ. "Tiến độ tụt hậu"" ngược lại trở thành kinh nghiệm trưởng thành cho mọi người.

TIẾN ĐỘ TỤT HẬU

Đã đạt tới cột mốc chưa? Thực ra vấn đề này rất khó có tiêu chuẩn cân đo hoàn toàn khách quan. Do thiết kế sản phẩm có thể thay đổi, mà bản thân chất lượng lại vô hình, do đó ngày đến hạt cột mốc cũng có thể thay đổi, vì vậy có đạt được cột mốc hay không lại

trở thành phán đoán của con người: khi càng ngày càng ít phải quay lại sửa lỗi, đạt được tôn chỉ của cột mốc, thành viên đều muốn thứ sức ở cột mốc tiếp theo thì bạn có thể tuyên bố cột mốc thành công. Đương nhiên, lúc này rất khó quyết định. Đối với vấn đề này, tôi chỉ có thể lấy một ví dụ phản diện để nói rõ quan niệm này, tôi chỉ có thể lấy một ví dụ phản diện để nói rõ quan niệm này: Chẳng hạn bạn hỏi đường, mọi người trả lời: "Anh đi thẳng, nếu thấy cái tháp nhọn tức là đã đi quá rồi". Nếu bạn đã đến gần cột mốc, nhưng hoạt động phát triển vẫn đang được tiến hành thì có nghĩa là bạn đã vượt rồi. Là lãnh đạo, bạn phải luôn nắm được mọi trạng thái của phần mềm, cần phải biết làm gì trong từng giai đoạn, và nên dừng các hoạt động nào.

Đã có lần, chúng tôi tiếp cận M1 nhưng mọi công việc đều rối tung lên, chúng tôi chỉ còn hai tuần, tình hình phần mềm vẫn rất dở: Số lỗi quá nhiều, tuy tỷ lệ lỗi của mỗi lập trình viên đều ở trong phạm vi hợp lý, nhưng tổng số lỗi cao đến đáng sợ, tình trạng số lỗi tăng lên mỗi ngày cho thấy phần mềm còn lâu mới đạt đến ổn định.

Chúng tôi đành phải tích hợp hàng ngày (Tham khảo nguyên tắc 32), tuy hơn một nửa chương chính kết hợp thành công, nhưng vẫn còn vài chương trình không được, trông có vẻ biên dịch và nối được nhưng không thể

kết hợp. Vấn đề này đã cản trở lớn đến công việc phía sau: kiểm thử và phân tích mã chương trình. Khi một vấn đề chỉ định được tìm ra một cách khó khăn để giải quyết, thường mất tối thiểu một ngày (Hãy ghi nhớ, vào giai đoạn của cột mốc, một ngày có thể tiêu hao 10% tài nguyên).

Thành viên và lãnh đạo không cảm thấy căng thẳng, khi tiếp cận cột mốc, thành viên đều biết mình đã cố hết sức nhưng không hiểu sao chẳng thấy có chút sức lực nào, tinh thần cột mốc hầu như không còn thấy nữa.

Trên thực tế, nguyên nhân chính của vấn đề này (Cũng là tình trạng thường gặp) là do kết hợp không hài hoà. Khi đưa một chương trình vào, sẽ thay đổi một chương trình khác, kết quả chương trình khác lại phát sinh vấn đề. Nhóm A dựa vào chương trình chung của nhóm B, nhóm A đã sửa và kiểm tra chương trình của mình nhưng lẽ ra nhóm B hỗ trợ phiên bản thứ N, họ lại sửa thành phiên bản thứ N+1, tuy sửa đổi rất ít nhưng lại đóng luôn phần của nhóm A, nhóm B cho rằng không ảnh hưởng đến chương trình của nhóm A nên không nói cho họ biết, kết quả chương trình của nhóm A không kết hợp được, vì lẽ ra nó phối hợp với phiên bản N của nhóm B, ai ngờ nhóm B lại đưa vào chương trình với phiên bản N+1.

Người lãnh đạo có cần ngăn chặn trường hợp này?

Hầu hết phụ trách dự án khi gặp trường hợp này, phản ứng đầu tiên là hành động bổ cứu - thành lập nhóm đặc biệt để tiếp quản vấn đề. Nhưng làm như vậy sẽ dẫn đến một số vấn đề: Nghiêm trọng nhất là cái giá cho hành động này rất cao, tuy làm như vậy thành viên sẽ chú ý đến trạng thái phần mềm và sự nguy hiểm - tôi coi hoạt động này là "Tiêu điểm mạnh" - bạn dùng quyền của người lãnh đạo để giúp thành viên đi đúng hướng. Phương pháp cảnh tỉnh này không nên dùng thường xuyên, nếu không nhóm sẽ quen ỷ lại vào lãnh đạo.

Nếu tình trạng tụt hậu thường xuyên xảy ra thì rất nguy hiểm. Tiến độ lạc hậu trở thành chuyện bình thường, lúc này sử dụng uy quyền của người lãnh đạo nâng cao cảm giác chịu rủi ro cho nhóm, nhưng phương pháp này chỉ thích hợp khi tinh thần sa sút, lúc này nhóm cần được kích thích, chính là hoạt động "Tiêu điểm mạnh" trong cùng một thời gian cho nhóm nhỏ làm vài dự án (thường là hai), nhưng mục tiêu của các dự án không liên quan với nhau.

Khi triển khai "Tiêu điểm mạnh", dù nhóm mạnh cũng ít nhiều cảm thấy lãnh đạo "Phản trao quyền", do đó mọi việc không thuận lợi đều coi như của lãnh đạo,

coi biện pháp này là không coi trọng quyết định của nhóm. Tâm lý phòng ngừa tăng, tâm tư bắt đầu bất ổn, thậm chí có người bắt đầu cáu giận. Dù lãnh đạo xuất sắc nhất khi sử dụng "Tiêu điểm mạnh" đều gặp phải một giai đoạn phản đối. Giáo dục không theo ý nguyện của đối phương, tuy có thể làm được nhưng rất gian khổ.

Đồng thời, bạn phải làm cách mạng với tâm lý phản đối của nhân viên phát triển xuất sắc nhất, có thể hoà hợp với họ không dễ, mà họ còn trực tiếp coi bạn như kẻ ngốc (Tham khảo nguyên tắc 4). Mặt khác, bạn cũng có thể phán đoán nhầm về mức độ tinh thần sa sút, lúc này sẽ rắc rối hơn, nhóm mạnh sẽ bật lại, nếu bạn khăng khăng không nhận sai sẽ làm mất sự tôn trọng và chân thành của họ đối với bạn, nếu không thì sự kiêu ngạo làm bạn đi càng xa hơn; Còn nếu nhóm không mạnh như vậy, họ không dám phản đối bạn, lúc đó bạn sẽ đi vào đường thất bại.

Khi bạn quyết định sử dụng "Tiêu điểm mạnh", các câu chuyện kể trên đều là giá thành và nguy hiểm của bạn. Giá thành cố định của bạn tối thiểu là vài ngày làm việc (khoảng một tuần), mà bạn cần phải nắm được sức tập trung của thành viên, duy trì hứng thú của họ, cho họ thấy là bạn làm thật, để họ nói thật cảm nhận tâm lý của họ, cho họ làm chút công việc thực tế.

Thời gian cần thiết cho giai đoạn khởi đầu này tuỳ theo độ lớn của nhóm, mong muốn hoà hợp và năng lực của nhóm.

Bây giờ chúng ta quay về các bước khi tiếp cận cột mốc đầu tiên, lãnh đạo yêu cầu nhóm có nhận thức rõ như thế nào? Làm gì để làm được như vậy? Sau đó sẽ có những hành động gì?

Trong trường hợp của chúng tôi, cần cho nhóm nhận rõ M1 đang đuổi sát, hơn nữa nếu không qua cửa ải M1 thì sẽ không thể hoàn thành sản phẩm; Chúng tôi cũng mong muốn để nhóm luyện tập làm phần mềm "hoàn chỉnh đợi tung ra", làm xong giao hàng; Sau cùng; chúng tôi mong nhóm trải qua hậu quả đáng sợ do hợp tác nhóm chưa chín muồi đem lại, từ đó học được rằng mỗi người trong chúng tôi đều có trách nhiệm như nhau, cũng như trách nhiệm tổng hợp - đó chính là đạt đến cột mốc M1.

Trong trường hợp của chúng tôi, biện pháp "Tiêu điểm mạnh" không thích hợp, do vấn đề của nhóm không phải là ý chí sa sút, mà lại phát hiện khi thấy vấn đề đã xảy ra, nhóm bật lại rất dữ dội.

Vậy phải làm gì?

Rất rõ ràng, chúng tôi không thể rót hết đầu tư vào M1, chúng tôi biết nhất định không để vấn đề kéo

dài. Hiệu ứng xấu do không thể tích hợp phần mềm đang lan rộng, biểu hiện là tỷ lệ lỗi quá cao. Động tác đưa chương trình vào càng cẩn thận, cần phải kiểm tra kỹ xem có ảnh hưởng đến các chương trình khác không, điều đó cũng có nghĩa là tốc độ giảm rất nhiều. Số lỗi vẫn rất cao, sửa đổi để xoá lỗi vẫn gây ra những vấn đề như ở phiên bản trước. Khi tình trạng không thể Tích hợp phần mềm kéo dài quá lâu, sẽ làm cho càng nhiều chương trình phải kết hợp vào phiên bản cũ, kết quả là việc kết hợp trở nên vô vọng. Chúng tôi rất quan trâm đến tình trạng này, cảm nhận rủi ro của thành viên luôn tỷ lệ nghịch với tiến độ công việc.

Hợp tác nhóm có tác dụng ảnh hưởng tương hỗ. Để hoàn thành công việc chỉnh thể, bạn phải chia thành công việc cục bộ. Có thể cho ba hoặc bốn người hợp thành một nhóm nhỏ, ảnh hưởng của các nhóm nhỏ sẽ lan ra toàn bộ nhóm, ngược lại, thất bại của nhóm cũng lan rộng, gây tổn hại cho phần mềm. Lúc này trọng tâm của bạn đặt vào nhóm nhỏ có vấn đề, điều này còn tốt hơn là muốn chỉnh đốn cả nhóm mà kết quả thu được chưa chắc ra sao.

Sau một đợt chẩn đoán và thử chữa trị, cuối cùng quyết định: Trước khi chưa thành công tích hợp hàng ngày, nghiêm cấm đưa các chương trình vào. Có ba nguyên nhân chính ép chúng tôi không làm không được:

Thứ nhất, không có động tác đưa phần mềm vào hài hoà, làm nhiều nhóm nhỏ khác bị ảnh hưởng, cuối cùng mọi người không biết phải làm thế nào; Thứ hai, kinh nghiệm của nhóm phụ trách kết hợp không đủ, vì nhiệm vụ kết cấu mới được chuyển từ phòng phát triển sang phòng bảo đảm chất lượng; Thứ ba, luôn có một nhóm không qua được xét nghiệm, là nguồn gây ra các vấn đề.

Tốt rồi, các triệu chứng trên cho chúng ta biết cái gì? Trạng thái tâm lý của nhóm rốt cuộc có vấn đề gì?

Xem xem nhóm có vấn đề gì?

Trong giai đoạn đầu tôi đã đưa ra lý luận này: từ phần mềm có thể nhìn ra trạng thái tâm lý của khách. Nhưng bây giờ sự việc không còn như vậy nữa rồi, nhóm có vẻ không thể phát huy hiệu năng công việc cần có. Lúc này bạn phải hỏi bản thân, những hành vi này của nhóm có lộ ra thông tin gì không?

Trước hết, chúng ta xem các vấn đề mà phần mềm không thể kết hợp được, động tác đưa chương trình vào không hài hoà cho thấy nhóm thiếu nhận thức có tính chỉnh thể: Giữa các nhóm nhỏ không trao đổi với nhau. Mục tiêu phát triển và viết chương trình hàng ngày đều thuộc trong nhóm, động tác phát triển chương trình giữa các nhóm không nhịp nhàng. Mỗi nhóm nhỏ đều đang tiến về mục tiêu M1, nhưng thành quả công việc

của họ không thể kết hợp thành sản phẩm. Cống hiến của một số thành viên cá biệt đều rất tốt nhưng bên ngoài nhóm thì không ai biết.

Vấn đề thứ hai là kinh nghiệm của nhóm kết hợp không đủ. Nhân viên đảm bảo chất lượng chưa làm bao giờ, trước đây đều là nhân viên phát triển khá tốt phụ trách. Những nhân viên này có đủ năng lực giải quyết các vấn đề thất bại trong kết hợp, do đó họ đã quen với các kỳ vọng quá cao về kết hợp. Họ cũng như "Chó giữ cửa" nắm giữ mấu chốt kết hợp, giải quyết bất cứ vấn đề gì, thậm chí chỉ dẫn các kiến thức là kỹ thuật phát triển chương trình. Còn nhân viên đảm bảo chất lượng mới bắt tay vào làm, không thể quan tâm chi tiết đến phần mềm như vậy, không có năng lực kỹ thuật để giải quyết các vấn đề thất bại trong kết hợp, đồng thời đây cũng không phải trách nhiệm của họ. Mặt khác, nhân viên bảo đảm chất lượng đang thử hoà nhập vào văn hoá nhóm, trước đây nhân viên phát triển là lớn nhất, nhưng trong nhóm bắt đầu xây dựng vai trò và mô hình làm việc cho nhân viên đảm bảo chất lượng, mới đầu nhân viên đảm bảo chất lượng không thể yêu cầu kiên quyết nhân viên phát triển phải sửa phần mềm đến mức nào.

Do đó, nhân viên phát triển cảm thấy thất vọng vì công việc kém của của nhóm kết hợp, vô hình chung

nhân viên phát triển thấy nhóm kết hợp chỉ là những người dốt, họ đã quen dựa vào các cao thủ thâm niên sửa chữa chương trình cho mình, những lập trình viên thiết kế chương trình này đã bị làm hỏng, họ không quan tâm đến quan hệ giữa chương trình thiết kế với chỉnh thể, cũng không nhận thức rằng chương trình của mình phải kết hợp với các chương trình khác mới được, ai cũng cho rằng mình chỉ quan tâm đến phần mềm của mình, kết hợp có thành hay không không phải việc của tôi. Hiện nay nhóm kết hợp không còn đi xoá lỗi cho mọi người, mỗi ngày đều có chương trình chất lượng cao là trách nhiệm của mỗi nhân viên phát triển. Công việc của họ là kết hợp chương trình tốt thành phần mềm chất lượng cao, bảo vệ các chương trình dự kiến và giám sát đốc thúc nhân viên phát triển sửa chữa phần mềm của mình. Trước đây nhân viên phát triển không tình nguyện làm các nhiệm vụ này, phản ứng của họ khi kết hợp thất bại là: "Hôm nay nhóm kết hợp không làm tốt". Nhưng bây giờ quan niệm của họ là "Chương trình hôm nay không tốt nên không thể kết hợp".

Vấn đề thứ ba là, luôn có nhóm không làm được, phần mềm của họ không thể kết hợp hay ổn định, luôn không đạt yêu cầu kiểm tra. Điều này không có gì bất ngờ, phần chương trình luôn bị bỏ qua, không có đủ nhân viên bảo đảm chất lượng, phụ trách dự án không đầu tư gì thêm, có lẽ do kết hợp thất bại mỗi ngày mà

gây ra bất công. Thực ra chương trình này rất quan trọng đối với phần mềm, chỉ có điều không hiểu tại sao nó luôn bị coi là cấp 2. Thế là chúng tôi phân chia lại tài nguyên, tìm đủ người có năng lực để thực hiện cần phải bồi đắp sự coi thường và đãi ngộ khác biệt trong quá khứ. Từ trạng thái tâm lý và bản chất sản phẩm, đều có thể nhìn rõ trước đây chúng tôi hạn chế một số chương trình như thế nào.

Tóm lại, nhóm không hài hoà, trách nhiệm phân chia không rõ ràng, hơn nữa không có trách nhiệm tương hỗ cho đối phương hay trách nhiệm chung về phần mềm, người quản lý cũng phân chia tài nguyên không thích hợp. Sau khi chẩn đoán rõ ràng, thu lại biện pháp sai "Tiêu điểm mạnh", bắt đầu hành động chữa trị đúng đắn.

Cho thành viên học cách phối hợp lẫn nhau

Rất rõ ràng, do nhóm thiếu nhận thức về nhiệm vụ chung, giữa các nhóm nhỏ vừa không trao đổi vừa không nỗ lực cho mục tiêu chung, do đó chúng tôi phải sử dụng hiệu ứng lan trong công việc nhóm. Nói cách khác, nếu bạn giải quyết vấn đề ở một chỗ nào đó thì cũng tương đương giải quyết vấn đề cho cả nhóm. Phụ trách dự án là nhà lãnh đạo và nhân vật trung tâm, anh ta phải có hình tượng quản lý tốt, phụ trách tìm kiếm tài nguyên, bảo vệ nhóm, dẫn dắt nhóm đi đến

thành công, đồng thời anh ta cũng là trưởng thực thi, trong con mắt người khác anh ta tương đương với tổng quát của phần mềm, sự việc của mỗi người cũng là sự việc của anh ta.

Phụ trách dự án hay cấp trên có yêu cầu nhóm phải duy trì sự hoà hợp trong một mức độ nào đó, cũng có thể dùng phương pháp khuyến khích để có hiệu quả hoà hợp tốt hơn. Nhưng nếu vừa bắt đầu nhóm đã đặt kỳ vọng vào sự hoà hợp chỉnh thể hiệu quả, mong muốn đó quá lớn, khả năng thất bại rất cao: Vì sự việc trong nhóm quá nhiều, quan hệ giữa các thành viên rất phức tạp hơn sẽ càng phức tạp khi quy mô nhóm tăng lên. Cách làm khá hay là, để vài phụ trách dự án bàn về tình trạng công việc, mục tiêu, hy vọng và kế hoạch kỹ thuật dài hay ngắn hạn của họ. Cách hoà hợp ở thứ bậc cao, số người ít, không che giấu tuy khó khăn, nhưng không phải không làm được. Sau đó để phụ trách dự án đi duy trì sự hoà hợp trong nhóm nhỏ của mình. Phương thức này giúp nhóm có thể nhanh chóng, tự nhiên xây dựng được sự hoà hợp chỉnh thể, tuy bạn không thể ra lệnh hay tạo dựng hoà hợp cho nhóm nhưng bạn có thể bồi dưỡng, có thể dẫn đầu.

Tuy có rất nhiều quản trị dự án, mỗi người phụ trách một lĩnh vực khác nhau của sản phẩm, nhưng trong đó có chuyên môn và chức quyền của ba người có

thể tổng hợp toàn cục: Một người trẻ nhất, nhưng kinh nghiệm phong phú, đã nhiều lần lãnh đạo công việc tung sản phẩm ra; Hai người khác có vẻ lớn tuổi, tuy chưa đảm nhận trọng trách như vậy nhưng cũng thuộc dạng xuất sắc. Hoà hợp của ba người rất kém, một trong những nguyên nhân là người trẻ nhất không dẫn dắt hai người kia để họ tiến bộ.

Còn có các nguyên nhân khác như ba phụ trách dự án này trực thuộc các cấp lãnh đạo khác nhau. Hơn nữa, trong giai đoạn đầu ra, các quản trị dự án chưa nắm được vai trò cần đảm nhiệm của mình, họ có xu hướng tuỳ biến, xem xem người đảm nhận công việc này trước đây làm gì thì bây giờ họ làm theo là được, hoặc họ cho rằng có một số việc đã được những người đi trước làm hết rồi, mình không cần phải bận tâm. Vấn đề này phản ánh trong việc Tích hợp phần mềm hàng ngày, chúng tôi phải phân tích các vấn đề thất bại trong kết hợp và mối quan hệ giữa ba người, đi sâu tìm hiểu mối quan hệ giữa hai vấn đề trên.

Đùn đẩy trách nhiệm

Nguyên nhân hay gây kết hợp thất bại là khi đưa chương trình vào, các phiên bản không hài hoà, kết quả tạo ra hiệu ứng dây chuyền càng khó kết hợp, nguyên nhân nữa là phương trình đưa vào chưa hoàn toàn chắc chắn là không có lỗi. Phụ trách dự án đương nhiên

không thể không biết nhóm đang sửa chương trình gì, phụ trách dự án (hay tối thiểu nhân viên đảm bảo kỹ thuật) cần gửi chương trình chưa thực sự hoàn chỉnh cho nhân viên phát triển sửa chữa, và theo dõi độ sửa. Chất lượng chương trình trước khi đưa vào chưa được coi trọng và nắm vững, khi chương trình không cẩn thận đưa vào kết hợp, hoặc kỷ luật phát triển chương trình không được tuân thủ, suy nghĩ bắt đầu tản mát và đùn đẩy trách nhiệm: "đều là do nhóm kết hợp làm không tốt".

Đùn đẩy trách nhiệm là hiện tượng thất bại trong hợp tác nhóm, nhất là trong môi trường cảm giác trách nhiệm quá thấp, hiểulầm sẽ dễ gây hậu quả. Những người hay nhóm có tiêu chuẩn đạo đức cao sẽ bị tổn thương trong đùn đẩy trách nhiệm. Đây là phản ứng phòng vệ xấu, chỉ cần có người bị trách mắng, mình thì không sao, trong trực giác mọi người đều muốn tìm "kẻ chết thay" để chứng minh mình không có vấn đề gì, mà không hiểu rằng tình trạng tổng thể đang ác hoá (Đó không phải là lỗi của tôi!). Đùn đẩy trách nhiệm là tìm cách đưa bản thân thoát khỏi vấn đề chung, coi vấn đề đơn giản hơn.

Đó chính là một trong những thói quen xấu nguyên thuỷ, không trưởng thành của con người, nhưng có tác dụng cải thiện trạng thái tâm lý của mọi người

trong thời gian ngắn. Đó là lý do tại sao hiện tượng đùn đẩy rất phổ biến, và thường có trong nhóm hợp tác ngắn hạn. Nhưng, đùn đẩy không phải là phương pháp lâu dài, tác dụng ngắn hạn không thực sự hữu ích, những người thông minh hay tự tin đều không muốn dùng biện pháp này, chỉ những người trẻ tuổi và sợ tổn thương mới làm như vậy.

Đáng buồn, đùn đẩy tắc trách là thói quen xấu, nó có thể lan rộng, cho đến khi hiệu năng công việc của mọi người sa sút đến mức không ai chịu được nữa mới thôi. Đa số người bị hại đều cảm thấy mình bị vơ nắm cả đũa, áp lực tâm lý sẽ làm họ có tâm lý phòng vệ lớn hơn và hành vi trả đũa. Mục đích của đùn đẩy tắc trách trốn tránh vấn đề kiểm điểm, để mặt các vấn đề tồn tại, tìm người gánh tránh nhiệm cho mình. Hậu quả sẽ ngày càng nghiêm trọng như băng tan, lan từ nhóm nhỏ đến người quản lý, rồi sau cùng là đòn chí tử đối với tổ chức.

Tính phá hoại của đùn đẩy tắc trách có thể chia thành hai tầng. Tình hình bị chỉ trích và coi thường làm họ tìm mọi cách để bảo vệ mình, hoặc tìm cách đẩy trách nhiệm đi, hoặc tìm người lãnh trách nhiệm thay mình. Dù trước đây hiệu năng công việc có tốt đến đâu thì cũng nhanh chóng bị sa sút. Một khi người thứ hai (Người bị đẩy trách nhiệm) cũng đối mặt với nỗi lo bị

chỉ trích, sẽ xuất hiện hậu quả xấu cho tổ chức: Thứ nhất, do đẩy trách nhiệm cho người khác dễ dàng như vậy, nên các vấn đề thực sự sẽ không được nghiên cứu và giải quyết, hơn nữa căn bản không có ai chú ý, họ đang bận tìm cách đẩy trách nhiệm đi, kết quả làm tăng nhanh căn bệnh đùn đẩy, ăn mòn dần hiệu năng làm việc của nhóm. Do đó, chỉ cần có một người vô trách nhiệm là sẽ khởi động vòng tuần hoàn đùn đẩy trách nhiệm.

Thứ hai, hậu quả có tính phá hoại càng thêm nghiêm trọng vì nội bộ đá trách nhiệm cho nhau, mối quan hệ đồng nghiệp bị phá hỏng, dẫn đến sự can thiệp của cấp trên. Kết quả phán đoán thường không làm cho mọi người thoả mãn, do đó khó có thể tìm ra phương pháp thích hợp để giải quyết, làm mất tín nhiệm giữa mọi người, độ trung thực giảm sút.

Bất cứ lãnh đạo nào đều cần có nhận thức chính xác về tình hình đùn đẩy tắc trách, vì đó là bệnh tâm lý của nhóm, dễ "lây lan", làm tổn thương nghiêm trọng đến hiệu năng công việc, lâu dài sẽ làm tổn thương cả nhóm.

Trong trường hợp của chúng tôi, phụ trách dự án cần có trách nhiệm xây dựng mục tiêu rõ ràng cho nhóm kết hợp, xây dựng kỳ vọng của cả nhóm đối với nhóm kết hợp: Bây giờ không còn nhân viên phát triển

cao cấp sửa chương trình cho mọi người nữa, thành công kết hợp phần mềm là trách nhiệm của mỗi nhân viên phát triển, mà trách nhiệm của nhóm kết hợp là trả lại các chương trình không đạt, và hỗ trợ nhân viên phát triển tìm ra các vấn đề mà không thể kết hợp.

Cuối cùng, để hiểu vấn đề này, chúng tôi đành phải kéo dài M1 vài ngày hay một tuần. Kéo dài hạn là điều khó tránh được, nhưng mượn việc kéo dài để cho các thành viên hiểu rõ nhóm của chúng tôi thật ra rất yếu, dù thời kỳ đầu đã giao ước, dù quản lý tin tưởng thành viên và không có biện pháp mạnh nào; để cho việc kéo dài lưu lại bài học khó quên trong lòng các thành viên. Bây giờ, chúng tôi phải có hành động ngăn chặn, chúng tôi cho dừng mọi động tác phát triển cho đến khi kết hợp thành công mới thôi. Hơn nữa, chỉ chương trình được thực sự kiểm tra mới được phép đưa vào.

Thế là chúng tôi cho các nhân viên phát triển hiểu, công việc của họ quan trọng như thế nào, nếu có bất kỳ lỗi nào đều có thể ảnh hưởng đến việc kết hợp phần mềm, do đó trước khi đưa vào phải đảm bảo hoàn toàn không có sai sót gì. Sau đó chúng tôi cử thêm người cho chương trình để có thể sửa chữa nhanh chóng, không còn ảnh hưởng đến việc kết hợp.

Phương thức hoà hợp và hiệu quả

Phương thức hoà hợp sự việc cũng như ám thị sự việc đó. Một đoạn email không nói lên việc gì quan trọng, cũng ít người chú ý đến nó; Còn họp trực tiếp lại là phương thức hoà hợp mạnh nhất, biểu thị sự việc được bàn bạc rất quan trọng, hoặc có nghĩa là không thể giải quyết chỉ dựa vào nhóm người chúng tôi và cấp trên mà cần có sự điều chỉnh.

Các vấn đề đưa chương trình vào phần mềm được thảo luận trong cuộc họp đặc biệt (Special war room meeting). Theo văn hoá của Microsoft, cuộc họp đặc biệt không giống các cuộc họp khác là chỉ thông báo, những người cần thiết đến đủ là được, cuộc họp đặc biệt có nghĩa: Thứ nhất, liên quan đến tung sản phẩm ra; Thứ hai, tối thiểu một giám đốc tham dự; Thứ ba, chủ đề thảo luận rất quan trọng, đồng thời phải có nhận thức chung; Thứ tư, tuy không có quy định về hình thức, tốt nhất là có đủ phụ trách các nhóm nhỏ tham gia, vì quyết sách liên quan đến nhiều vấn đề. Thông thường giám đốc dự án cao nhất triệu tập và chủ trì cuộc họp này, nhưng các lãnh đạo khác cũng có thể triệu tập cuộc họp đặc biệt.

Cần chú ý là cuộc họp đặc biệt khác hẳn với cuộc họp truyền đạt mệnh lệnh, nếu giám đốc dự án nói chúng tôi cần làm gì, phải như thế nào, sau đó mọi

người sẽ làm như vậy. Tuy cũng thảo luận, nhưng cơ bản là trong phạm vi cho phép của người chủ trì. Còn truyền đạt mệnh lệnh, sẽ không có chỗ cho thảo luận, thảo luận chỉ có tác dụng hiểu rõ hơn về quyết sách. Nếu trong cuộc họp có người thông minh hơn, hiểu vấn đề hơn người ra quyết sách, hoặc có biện pháp hay hơn, anh ta sẽ cảm thấy buồn chán, những người khác không được dùng biện pháp hay hơn nên cảm thấy thất vọng. Còn trong cuộc họp đặc biệt, các quyết sách đưa ra sau thảo luận thường là những quyết sách thông minh, khi thực hiện cũng rất có hiệu quả. Thông thường có người phát hiện vấn đề lớn, đồng thời suy nghĩ kỹ, hội ý kiến chuyên gia, có được những phương hướng hành động sơ bộ, trong cuộc họp đề nghị mọi người đưa ra cách nghĩ của mình. Những người tham dự đã có hiểu biết nhất định về vấn đề này trước khi họp, thậm chí đã suy nghĩ hướng giải quyết, nếu vấn đề đó chưa được nhắc đến thì có thể tự đưa ra bàn bạc.

Trong những cuộc họp này, các ý kiến hay kiến nghị đưa ra đều đã được suy nghĩ kỹ, các phương hướng sơ bộ sẽ được bổ sung, sửa đổi, nhận thức chung được hình thành trong quá trình này. Nếu ai có ý kiến thì cần cho họ hiểu rõ thêm, sau đó khi mọi người có chung ý nghĩ về vấn đề, sẽ đồng tâm giải quyết.

NGUYÊN TẮC 43: KHÔNG SỬA KỲ HẠN CUỐI CÙNG CHỈ VÌ TIẾN ĐỘ TỤT HẬU

Don't trade a bad date for an equally bad date

Sau này khi nghỉ hưu, tôi rất muốn nghiên cứu ra công thức số học về tiến độ tụt hậu. Hiện nay tôi có thể khẳng định, tiến độ tụt hậu đầu tiên thường nghiêm trọng nhất, sau đó do tích luỹ kinh nghiệm mà bạn nắm tiến độ tốt hơn, vì vậy khi lần đầu tiên bị tụt hậu, bạn phải kiên nhẫn.

Mức độ tiến độ tụt hậu tỷ lệ thuận với tính bất định của kế hoạch. Cũng như bạn giở một cuốn sách gọi là cuốn sách tương lai, bạn không thể dự liệu tương lai sẽ ra sao trừ khi bạn tiến tới thời điểm đó; Cùng với thời gian, các vấn đề chưa xác định đều trở thành được biết, do đó tính bất định của kế hoạch ngày càng giảm, tiến độ trở nên dễ nắm bắt. Về lý luận, mức độ tiến độ tụt hậu tương đương với độ bất định của kế hoạch, không phải như số học giảm về không nhưng không bằng không, do đó nếu mức độ bất định là 0, sẽ không thể có tụt hậu.

Tôi tin rằng cần phải có cách tính sau: Dùng trục thời gian (X) và bất dịnh (Y), cho hai giá trị bất kỳ về tiến độ bất định, sẽ có được ngày hoàn thành dự án. Tôi

tin vào cách tính này, vì lần nào tôi cũng dùng nó, tuy tôi không thể đưa ra công thức chính xác hoặc chứng minh. Nhưng bạn hãy mở to mắt chú ý, tiến độ tụt hậu mà bạn trải qua có phải ngày một nhẹ hơn, nếu vậy là tốt, nếu không bạn đang xa dần mục tiêu.

Trước khi bạn biết được lúc nào mình có thể hoàn thành, bạn có thể phải trải qua những lần tụt hậu.

Nhưng, không được thay đổi thời hạn cuối cùng để giảm bớt tiến độ tụt hậu, nếu làm không tính toán bạn sẽ bị mất hết. Thông thường, trước khi bạn biết được lúc nào mình có thể hoàn thành, bạn có thể biết rằng dự án đã kéo dài, lúc này các thành viên trong nhóm hay cả công ty, kể cả phóng viên báo chí (nếu họ biết được) đều có thể hỏi bạn: "Nếu ai cũng biết rằng không thể tránh khỏi kéo dài tiến độ, vậy tại sao không dịch chuyển kỳ hạn cuối cùng?" Dù có nhiều áp lực ép bạn phải gia hạn, thì ngày đáo hạn chính thức vẫn là ngày cuối cùng của dự án, nếu tùy tiện thay đổi, các thành viên sẽ cho rằng thời hạn trước đây được đặt không thoả đáng, làm mọi người dao động. Ngày đáo hạn của dự án không phải không sửa được, nhưng không nên sửa chỉ vì tiến độ không đạt.

Khi tiến độ tụt hậu, biện pháp đáng sợ nhất là kéo dài thời hạn cuối cùng. Điều đó chẳng khác gì mang một việc bạn làm sai (Tiến độ tụt hậu) tiếp tục vác đi (Tiến độ có thể tụt hậu một lần nữa); Phương pháp này trông có vẻ đơn giản (Chúng ta không cần phải nghĩ đến nó), nhưng lại rất ấu trĩ. Lúc này vấn đề mà bạn cần làm là suy nghĩ xem cái gì dẫn đến tiến độ tụt hậu.

Tuy thông tin bạn biết bây giờ phong phú hơn rất nhiều so với khi thiết lập kỳ hạn trong giai đoạn đầu, nhưng vẫn chưa đủ để bạn quyết định lịch trình mới. Ngược lại, muốn nói rõ trong trường hợp nào có thể đặt lại kỳ hạn là vấn đề rất khó. Nguyên tắc thông thường khá tốt là, trừ phi bạn đã biết các chương trình kết hợp gây tụt hậu đến mức nào, còn thiếu những nội dung gì, đã thực sự tìm ra nguyên nhân tụt hậu và đang khắc phục vấn đề, nếu không thì tuyệt đối không được sửa đổi ngày hạn cuối cùng. Đương nhiên để làm được như vậy, cần có sự hợp tác của cả nhóm, lãnh đạo chính xác ra kỳ hạn (tôi định nghiên cứu sau khi nghỉ hưu), bạn không thể tính ra kỳ hạn chính xác. Bạn cần đặt mục tiêu trên cột mốc gần nhất một cách thiết thực, đồng thời chuẩn bị tâm lý, đường đi sau này còn nhiều nhân tố bất định.

Trân trọng thời gian của bạn, sử dụng hợp lý để tìm ra nguyên nhân tụt hậu chứ không tính xem nên

kéo dài bao lâu, bạn không thể do mất thời gian giải quyết vấn đề mà làm lỡ càng nhiều thời gian, ngược lại, bạn sẽ làm nhóm phát triển ngày càng ổn định, sớm đạt đến mục tiêu.

NGUYÊN TẮC 44: LỠ CỘT MỐC NÀY, NHƯNG PHẢI ĐẠT ĐƯỢC CỘT MỐC SAU

After a slip, hit the next milestone, no matter what

Chúng ta cần biết rằng, mỗi lần bị kéo dài là một lần lòng tin của bạn và cả nhóm bị giảm sút, do đó khi lỡ một cột mốc, cách bổ cứu tốt nhất là bằng mọi cách không để lỡ cột mốc tiếp theo. Nhóm phải cứu vớt lại lòng tin của mình và giao ước với lý tưởng, vì vậy ý nghĩa hoàn thành đúng hạn của nhiệm vụ tới rất lớn, nhóm cần xây dựng lại lòng tin của mình.

Bạn biết được tiến độ tụt hậu nghiêm trọng đến thế nào, cũng biết được nguyên nhân tạo thành, bạn biết cách giải quyết vấn đề, sau đó xây dựng cột mốc tiếp theo khá gần và có thể tính toán được, lần này không được để lỡ, sau đó thông báo thông tin này. Nếu lịch trình của bạn là ba tuần tới không làm gì thì thật không kịp. Ý của tôi là bạn phải xây dựng cột mốc có thể hoàn thành, dù cột mốc này không có nội dung gì đặc biệt, và bắt buộc phải đạt được mục tiêu, động tác này phải kiên quyết có sức mới được.

Tuyệt đối không để lỡ lần nữa, vì nhóm cần mượn cơ hội này để xây dựng lại lòng tin. Đạt đến cột mốc đúng hạn, mục tiêu thành công, sẽ cổ vũ tinh thần cả

nhóm, tràn đầy sức sống, tin tưởng mình có khả năng đạt được lịch trình. Tâm tư này rất quan trọng, nếu nhóm tin rằng mình đạt được cột mốc, họ sẽ cố gắng hết sức mình, đó cũng như phương thức làm việc cần có của con người.

NGUYÊN TẮC 45: COI TỤT HẬU LÀ CƠ HỘI HỌC TẬP

A good slip is a net positive

Coi tiến độ tụt hậu như kinh nghiệm học tập: Bạn đã từng không hiểu vấn đề, bây giờ hiểu rồi, lúc đó là cơ hội tốt nhất để phân tích, suy nghĩ. Những tâm đắc học được lúc này là sâu sắc nhất, có lẽ rất khó dùng từ ngữ miêu tả. Nếu bạn dùng ấn tượng cứng nhắc để xem xét tiến độ tụt hậu, coi đó là việc xấu, thì thực tế của tiến độ tụt hậu không làm cho tiến bộ mà trở thành sự việc đáng sợ hay phát sinh của bạn.

Tuy có rất nhiều nhân tố bất định gây ra tiến độ tụt hậu, nhưng việc kéo dài đã phát sinh và nhiều khi còn coi như hiện tượng thông thường. Có rất nhiều công việc phát triển phần mềm về bản chất rất có tính thực nghiệm; Hệ điều hành mới, kỹ thuật mới,... làm cho mỗi dự án đều đầy tính bất định.

Việc kéo dài tuy không tránh khỏi, nhưng để tránh phát triển thành nỗi hoạ lớn, bạn phải biết đo lường các nhân tố gây tụt hậu. Tình hình lý tưởng nhất là, bạn đã biết hơn một nhân tố bất định, và cho nhân viên biết bản thân tiến độ cần có những rủi ro này, cho họ thấy cách đánh giá ảnh hưởng của các nhân tố này đối với tiến độ, cách tính thời gian mà rủi ro tồn tại,

năng lực dự tính này là kiến thức và kỹ thuật của nhóm, rất hữu ích trong tương lai. Đồng thời bạn phải biết cách phán đoán xem mọi người có đang làm "đúng không", nguyên nhân thường có của tiến độ tụt hậu là, mọi người tốn nhiều thời gian nghiên cứu tính năng bên ngoài hoặc phát sinh, thực tế nó không có tác dụng đối với thông tin trọng tâm của sản phẩm.

Nếu tụt hậu chỉ là đột nhiên xuất hiện, điều đó có nghĩa là con đường hoà hợp không thông suốt, bạn phải tăng năng lực hoà hợp không thông suốt, bạn phải tăng năng lực hoà hợp cho nhóm, cung cấp hàng loạt thông tin cho các thành viên để họ lĩnh hội vấn đề một cách cụ thể mà chân thực, tụt hậu bộc lộ ra điểm yếu của nhóm, nhưng cũng là cơ hội để loại bỏ các yếu điểm này. Bạn phải chắc chắn rằng vài trò của mỗi người đều được chỉ bảo.

Tụt hậu cũng là cơ hội tốt để đánh giá tương lai tương đương với mục tiêu gì, cần có những tài nguyên nào, tính năng sản phẩm cần có sau này sẽ được lựa chọn một cách thiết thực, các tính năng bất lợi cho hiệu năng của nhóm sẽ không được đưa vào mục tiêu, tác dụng tương hỗ giữa tài nguyên và tính năng cũng có thể được tính toán chính xác.

Tóm lại, có thể làm cho nhóm học được từ tụt hậu chắc chắn sẽ là việc tốt.

NGUYÊN TẮC 46: NHÌN CÂY THẤY RỪNG

See the forest

Nếu dự án có 6 *mẫu*, trong đó mỗi *mẫu* đã hoàn thành được 90%, vậy có nghĩa là bạn đạt được 45%. Mỗi *mẫu* đạt 90% xem như là thành tích khá tốt, nhưng không được coi như cả dự án đạt được 90% bởi chúng không liên quan với nhau. Bạn phải "nhìn cây mới thấy rừng". Nếu bất kỳ một mẫu nào có tỷ lệ hoàn thành là 0, thì tỷ lệ hoàn thành của cả dự án cũng là 0.

Dùng phương pháp số học này để tính toán tỷ lệ hoàn thành của cả dự án là rất chính xác, vì mỗi bộ phận đều có tính quan trọng tương đương trong sản phẩm, giả sử nhóm được phân chia đều công việc thì bất kỳ một bộ phận nào thất bại cũng là thất bại của cả nhóm.

Điều đáng chú ý là, không có thể nắm chắc 100% bất cứ sự việc nào, nếu có thì rất dễ gây ra hậu quả "kiêu tất bại". Anh hùng hôm nay có thể thất cơ lỡ vận vào ngày mai, tôi luôn kinh ngạc về sự trượt dốc không phanh của các ngôi sao nổi tiếng. Bạn phải cho thành viên hiểu rằng giành được danh dự không dễ, nhưng lại dễ vỡ. Nhóm phải luôn nghiêm khắc với bản thân, không được tuỳ tiện coi thường người khác, không kiêu căng ngạo mạn, không để nổi lo trong lòng đánh thắng bản thân.

NGUYÊN TẮC 47: THẾ GIỚI THAY ĐỔI, BẠN CŨNG PHẢI THAY ĐỔI THEO

The world changes, so should you

Công việc phát triển phần mềm thành công có một đặc tính rất quan trọng, đó là có thể đưa ra quyết sách đúng đắn trong hàng loạt thông tin hàng ngày. Bạn không cần quá câu nệ vào kế hoạch hay tiến độ, đó là sự thật nhân tạo, không tránh khỏi có lúc không thật. Thay đổi là mẹ của cơ hội, nếu bạn không sử dụng linh hoạt các kiến thức học trong trường, bạn sẽ mất đi nhiều cơ hội, rất khó thích ứng với môi trường đang thay đổi chóng mặt, bạn sẽ bị thay đổi hay cơ hội biến thành cản trở, phiền nhiễu cho kế hoạch hay tiến độ, chứ không phải là cơ hội biến đổi đầy tiềm lực.

Phát triển phần mềm cùng như thể hữu cơ không ngừng thay đổi, thông thường một khó khăn lớn cũng có nghĩa là mong muốn làm ra phần mềm tốt của nhóm. Trước khi có phản ứng trực giác, trước khi muốn giải quyết vấn đề, hãy bỏ chút thời gian tìm hiểu các cơ hội phía sau trạng thái tâm lý này, thử đưa năng lực vào đúng hướng. Bản thân vấn đề có thể biểu đạt rất nhiều ý nghĩa, bạn phải vận dụng khả năng quan sát, tưởng tượng và giác quan thứ sáu để quyết định làm gì

để thông suốt quá trình phát triển phần mềm. Luôn ghi nhớ bạn là tâm điểm lãnh đạo mọi người, cùng tiến hành công việc có tính sáng tạo. Nếu tính dự đoán càng cao, có nghĩa là tính sáng tạo càng thấp, do đó cần đảm bảo sức bật cho mỗi sự việc.

Đương nhiên, bạn không mong muốn thay đổi dễ dàng, bất cứ thay đổi nào đều không làm bạn xa rời phương hướng, thay đổi quá lớn làm mọi người không theo kịp. Sức bật không có nghĩa là tuỳ tiện, sức bật là tính thích ứng, là biến đổi tự nhiên; Còn tùy tiện là thay đổi đột ngột, hỗn loạn không liên quan. Nếu thế giới bên ngoài không thay đổi, thay đổi tuỳ biến một chức năng có thể đem lại điều tốt, nhưng duy trì nguyên hiện trạng sẽ an toàn hơn. Chỉ chấp nhận thay đổi khi có thể tăng hiệu năng chỉnh thể, mục đích thay đổi là làm công việc của nhóm trở nên tốt hơn, chứ không phải đưa vào càng nhiều phức tạp.

Chúng ta ví dụ thế này vậy, có một ngày, bạn muốn tăng thêm một cột mốc, cần phải có lý do rõ ràng để làm như thế này, nhưng có những dấu vết xác định đó là hướng phát triển trước mắt, lúc này tăng cột mốc có thể tăng rủi ro tụt hậu, nhưng cũng tăng cơ hội sau khi đưa sản phẩm ra. Tôi đưa ra lý do này là vì gần đây nhóm của chúng tôi thường xuyên gặp phải khó khăn tương tự, tuy chúng tôi thường xuyên gặp phải các khó

khăn tương tự, tuy chúng tôi luôn tự hào vì có thể tung sản phẩm ra đúng hạn, nhưng thay đổi của môi trường làm chúng tôi ngày càng nhận thấy cần phải tạo ra sự khác biệt, cần có đột phá nên đã đưa một cột mốc mới vào kế hoạch cũ để tạo ra một số đặc biệt. Mỗi dự án đều như trẻ mới sinh, mỗi dự án đều độc đáo, bạn cần thuận theo tính chất độc đáo của dự án để đưa ra cột mốc thích hợp, và hỗ trợ đầy đủ, dùng ngôn ngữ của nó để biểu đạt, chứ không như bánh gatô cắt thành hình bạn muốn.

Tuy bạn muốn làm một số thay đổi, bạn chưa chắc có dũng khí thực hiện.

Phần mềm lớn sẽ chỉ có một tư tưởng trung tâm, còn việc thực hiện chất lượng đến mức độ nào, còn tuỳ thuộc người lãnh đạo có thể dẫn dắt nhóm hoà hợp vào hàng loạt các thay đổi nhỏ nhưng quan trọng. Nếu bạn có thể nhận ra những thay đổi có ý nghĩa đối với dự án từ trong hỗn loạn, và dẫn dắt nhóm thích ứng, đưa chúng vào tinh thần nhóm, sau này trong sản phẩm sẽ biểu hiện các thay đổi này, bày ra trước mặt khách hàng.

Chống lại thay đổi là sách lược sai lầm, bạn phải tìm ra những thay đổi không thể chống lại hoặc có lợi

cho sản phẩm, bảo vệ và thích ứng với những thay đổi đó, cuối cùng chúng sẽ đem lại cho bạn rất nhiều thu hoạch. Đây là con đường khó, bạn cần có ý chí sắt thép vì bạn phải dựa vào sự nhìn xa trông rộng của mình, những gì bạn thấy người khác chưa chắc đã thấy, có thể bạn rất cô đơn, nhiều đêm bị nghi ngờ dày vò mà không ngủ được. Tuy bạn rất muốn làm một số thay đổi, nhưng chưa chắc đã có đủ dũng khí thực hiện, bạn giống như đang thách thức kỳ vọng của mọi người, hơn nữa cảm giác của bạn cũng vì thế mà bị dao động, bạn không dám khẳng định những thay đổi đó là tốt. Bạn có thể sẽ đối mặt với một chọn lựa khó khăn, bị giày vò giữa thay đổi và không thay đổi.

Nhưng dù thay đổi có lực cản thế nào, chỉ cần là những thay đổi bắt buộc, bạn phải loại bỏ mọi khó khăn để tiếp nhận nó, nếu không bạn sẽ bị nó thiêu huỷ.

Đối với tôi, thực ra chỉ cần viết được những ý nghĩa như vậy đã làm tôi cảm thấy kinh ngạc sợ hãi, nhưng đó thực sự là những kinh nghiệm lĩnh hội được trong quá trình phát triển phần mềm.

PART Three

PHẦN 3
GIAI ĐOẠN HOÀN THÀNH

Đối với nhóm phát triển phần mềm, không có gì có thể làm mọi người hưng phấn bằng việc sản phẩm chuẩn bị tung ra. Đa số cho rằng sau khi sản phẩm trải qua giai đoạn này, có thể nói là ra đời, nhưng tôi vẫn có khuynh hướng chia thời kỳ này thành 3 giai đoạn: kích hoạt, di chuyển và hoàn thành. Ba giai đoạn nhỏ này không có ranh giới rõ rệt, cũng có phần chồng lên nhau, nhưng trọng điểm công việc của chúng khác nhau mà nhóm phát triển nào cũng phải lần lượt trải qua mới có thể tung ra sản phẩm. Phương thức phân biệt ba giai đoạn này của mỗi người cũng có thể không giống nhau, nhưng đối với cả quá trình trong sản phẩm ra, ai cũng có cách nhìn và cảm thụ giống nhau.

TUNG SẢN PHẨM RA: KÍCH HOẠT

Kích hoạt là một quá trình dài mà tiệm tiến, cũng như cơn đau trước khi sinh nở. Lúc này nhóm bắt đầu các công việc chuẩn bị phức tạp cho việc tung sản phẩm ra, chẳng khác gì trước khi đẻ sản phụ phải tập hít thở, các chức năng cơ thể cũng phải có chuẩn bị thông thường đối với thời điểm sinh. Vào thời điểm tung sản phẩm ra, cả nhóm phải tập trung tinh thần và sức lực để nó thuận lợi ra đời, thực tế là thời điểm căng thẳng mà đau khổ.

Giả sử chúng ta có 4 hoặc 5 cột mốc, đại khái đến

M3 sản phẩm đã được hình thành, đạt đến M3 chính là "kiểm tra mô phỏng" lần thứ nhất tung sản phẩm ra. Cột mốc thứ tư hay thứ năm sau đó, cùng nhấn mạnh luyện tập trước khi đưa sản phẩm ra, không chỉ bản thân sản phẩm ngày càng tiếp cận hình mẫu, việc đạt đến cột mốc cũng giống với việc đưa sản phẩm ra.

Thông thường điềm báo đầu tiên bước vào giai đoạn kích hoạt là, một số thành viên phải suy nghĩ cẩn thận bắt đầu lo lắng: "Chúng ta thực sự làm được không?". Điều này không có nghĩa là nhóm sợ rằng mình không làm được, mà là một số người bắt đầu suy nghĩ hàng loạt việc phải làm, hoặc những khó khăn đang đến gần, nghĩ đến nhiều công việc phức tạp cần phải làm, cảm thấy trăm mối tơ vò, không biết thế nào mới tốt, nhất là khi số người suy nghĩ còn ít. Dần dần, lo lắng thậm chí kinh ngạc lan dần ra cả nhóm, có lẽ lãnh đạo cũng lộ ra những tâm tư như vậy ở nơi công khai. Khi cả nhóm có tâm tư căng thẳng này, lãnh đạo lại thở dài nhẹ nhõm, vì lúc này ai nấy đều cùng quan tâm một vấn đề, tung sản phẩm ra là ý nguyện mạnh mẽ của mỗi người, ai cũng có cảm giác đồng lòng, cảm giác nỗ lực chung cho một sản phẩm lớn, cảm giác trải qua bao khó khăn phát triển cuối cùng đã có kết quả.

Có một số thành viên hoài nghi, lãnh đạo đã quá lạc quan.

Bây giờ, việc lãnh đạo cần làm là cổ vũ mọi người, rót thêm nhiệt tình và tự tin, có một số thành viên cho rằng lãnh đạo đã vui quá sớm, nhưng đa số mọi người đều được khuyến khích, hiểu rằng những lo lắng lúc này là bình thường, là phản ứng tự nhiên có thể chấp nhận được, và là áp lực phải trải qua trong quá trình xây dựng sản phẩm trí tuệ.

Hiệu suất làm việc của nhóm trong giai đoạn kích hoạt cũng đạt đến trạng thái cao nhất, vì xu thế đã định, không cần phải bàn bạc. Lúc này tâm lý nhóm có vài đặc điểm sau:

- ***Thế lớn đã định:*** Tính năng sản phẩm, vai trò mỗi người trong nhóm hầu như đã hoàn toàn định hình. Những do dự ban đầu đã được loại bỏ, gánh nặng công việc giữa các thành viên rất đều, khung sản phẩm đã xác định và có xu hướng chín muồi, kế hoạch kiểm thử đang được tiến hành, đã có đại cương tài liệu, bây giờ đang viết chi tiết. Lúc này không có tranh luận, hay nghi ngờ, ai cũng biết sản phẩm sẽ ra đời, những việc còn lại cũng rất rõ ràng, chỉ cần đi làm là được. Công việc phát triển khó khăn cũng qua đi, lúc này nhanh chóng hoàn thành là quan trọng nhất.

- ***Niềm tin chung:*** Ai cũng có niềm tin chung, chúng ta sắp hoàn thành sản phẩm, đến thời gian là đưa sản phẩm ra. Những lựa chọn khó khăn đã qua đi,

mọi người chỉ chú ý vào trạng thái sản phẩm trước mắt. thời kỳ trợ sản của sản phẩm có nghĩa là mọi tranh luận đã kết thúc. Nếu vẫn còn nghi ngờ hay tranh luận (đến bây giờ vẫn chưa giải quyết) thì chỉ còn cách gác lại, sẽ bàn đến trong phiên bản tiếp theo, nếu không sẽ không bao giờ hết. Lúc này nếu vẫn có người nghi ngờ nghiêm trọng về dự án, sản phẩm sẽ không bao giờ vào được thời kỳ trợ sản.

- ***Ai cũng rõ ràng:*** Ai cũng rõ mình nên làm gì, cũng rõ nhóm cần làm gì mới có thể làm sản phẩm tung ra. Bây giờ đã không còn việc gì bất định. Do vậy, chỉ cần tiếp tục làm công việc của mình là được. Kéo dài hoặc lo lắng bị kéo dài chỉ đơn thuần liên quan đến công việc của bản thân, không còn liên quan đến công việc của người khác. Các hiện tượng tâm lý như lo lắng, bất an giờ đã được loại bỏ. Tôi thường gọi giai đoạn này là "Bảng kê công việc", mỗi nhóm nhỏ đều lần lượt hoàn thành và đánh dấu, khi làm hết công việc trên bảng kê, sản phẩm hoàn thành và được đưa ra.

Lãnh đạo cao nhất của nhóm phát triển phải là lính tiên phong, dẫn dắt trạng thái tâm lý cần có khi vào giai đoạn này, đồng thời tin tưởng mọi giao ước mới có thể đưa mọi người vào thời kỳ căng thẳng này. Nhận thức về các chi tiết cũng ngày một rõ ràng, cũng có đôi chút rủi ro hỗn loạn trên bước đi, nhưng lúc này điều

quan trọng nhất là dẫn dắt mọi người tập trung vào việc tung sản phẩm ra, các công việc không liên quan đều được loại bỏ, mọi thành viên đều muốn đưa sản phẩm ra, nhìn thấy kết tinh thành quả với cống hiến khó khăn của mình, lãnh đạo cần hướng khát vọng vào việc đưa sản phẩm ra, chuyển hoá thành công việc đưa sản phẩm ra.

TUNG SẢN PHẨM RA: CHUYỂN TIẾP

Trong giai đoạn này, về thời gian có thể vượt hơn một cột mốc tuỳ theo nhu cầu của tính chất sản phẩm và việc đưa sản phẩm ra. Về cơ bản, đây chính là giai đoạn dần dần ổn định của sản phẩm từ lúc phát triển đến hoàn thành. Công việc phát triển vẫn tiến hành, cho đến cột mốc cuối cùng mới thôi. Nếu trọng tâm công việc phát triển là tăng thêm chức năng, rất khó cùng lúc mong muốn sản phẩm ổn định, tự nhiên sẽ ảnh hưởng nghiêm trọng đến việc đưa sản phẩm ra; do cùng lúc tìm kiếm tăng trưởng và ổn định rất hao phí sức lực, hoạt động phát triển chủ yếu lúc này là loại lỗi, các tính năng chưa hoàn thành được đều cần gác lại. Muốn gác lại các tính năng, sẽ không tránh khỏi việc xuất hiện tâm lý bất an của thành viên, do đó trước khi quyết định cần phân tích cẩn thận.

Nếu chúng ta coi đưa sản phẩm ra như một đường gấp khúc liên tục trong toán học, bạn sẽ thấy khi sản phẩm ngày càng chín muồi, nhân lực cần thiết càng ít, sản phẩm đưa ra sẽ tiến vào giai đoạn hoàn thành (lát sau chúng ta sẽ thảo luận kỹ). Bây giờ, giai đoạn di chuyển chuẩn bị kết thúc, còn khi tiến vào giai đoạn hoàn thành thời kỳ đầu sẽ có những dấu hiệu sau:

- Suy nghĩ của thành viên càng bảo thủ và cẩn thận, để tránh sa lầy.

- Động tác đưa chương trình vào rất thận trọng, thường thành lập một nhóm nhỏ chuyên cần tạm thời, kiểm tra mỗi động tác đưa vào, nhằm ngăn ngừa những ảnh hưởng bất lợi có phạm vi lớn.

- Trọng điểm công việc là tăng tốc độ thực thi và cải thiện giao diện người dùng, đánh bóng sản phẩm, đem lại ấn tượng tốt.

- Hoàn thành công việc trang trí, tiến đến là tăng tính ổn định cho sản phẩm nhằm đạt được tiêu chuẩn tung sản phẩm ra, các công việc phát triển đã kết thúc, công việc loại lỗi cũng gần kết thúc.

ĐƯA SẢN PHẨM RA: HOÀN THÀNH

Trong ba giai đoạn đưa sản phẩm ra, trọng tâm công việc thay đổi, tâm lý thay đổi từ thời kỳ chuẩn bị

đến trợ sản, sản phẩm dần ổn định, cho đến giai đoạn hoàn thành đưa sản phẩm ra, chỉ còn một cột mốc nữa. Về khái niệm, công việc của giai đoạn này khá đơn thuần, chỉ là bảng liệt kê công việc, khi các công việc liệt kê dần được hoàn thành, sản phẩm dẽ được đưa ra. Bảng kê công việc có mấy trăm mục, thậm chí hàng ngàn mục thì cũng chỉ là một bảng kê, rất nhiều nhưng không khó, bạn không có thời gian nghĩ hoặc làm các công việc bên ngoài bảng kê, ai cũng chủ tâm vào các việc cần làm của mình. Cũng có thể phát sinh một số tình huống làm phải bổ sung thêm một số mục vào bảng kê, sau một trận phản đối nhóm sẽ chấp nhận việc bổ sung hạng mục, loại phản đối này, tôi gọi là "phản đối khiêm tốn".

Lúc này mỗi ngày đều họp kiểm điểm là phương thức khá tốt, nếu thường xuyên mời được người tham gia quyết sách càng tốt. Cuộc họp không câu nệ vào thói quen hay bất cứ hình thức nào, nhưng cần phải có năng lực quyết đoán vấn đề, các mục tiêu lâu dài hay những vấn đề không thể giải quyết ngay đều tạm thời gác lại không bàn, để đến phiên bản tới sẽ nhắc đến. Một trong những tác dụng khi người quản lý tham gia là dẫn chủ đề theo hướng này. Hiện giờ, mục tiêu chính của nhóm là đưa sản phẩm ra đúng ngày, chất lượng sản phẩm phải đủ tốt. Đã bước vào giai đoạn đếm ngược, công việc chính của nhân viên phát triển là sửa

lỗi, mà đó là những lỗi quan trọng - nó ảnh hưởng lớn đến người sử dụng, hoặc gây ra tính phá hoại nghiêm trọng, những lỗi đó phải ưu tiên xoá bỏ. Còn việc làm đẹp giao diện người dùng, tăng tốc độ thực thi, hoặc chức năng mới đều không nên thảo luận trong giai đoạn này. Kết quả thử nghiệm bản Beta, phản hồi của mọi người đều dựa trên tiền đề "sự ổn định sản phẩm và tránh lỗi nghiêm trọng" để sửa đổi nhỏ. Còn các kiến nghị khác, chỉ là cảnh báo các khó khăn hay nhân tố bất lợi trong khi tiêu thụ, hoặc là ý kiến cải tiến trong phiên bản tới, nhưng về cơ bản, trước mắt không đưa các hạng mục này vào sản phẩm.

Có bao nhiêu lỗi không ảnh hưởng đến đại cục, tồn tại trong sản phẩm của bạn.

Bạn cần phải hiểu, mức độ yêu cầu của khách hàng đối với chất lượng phần mềm hay mức độ chấp nhận lỗi. Trong sản phẩm đưa ra, có bao nhiêu lỗi do được ưu tiên sửa mà giảm đi, còn lỗi chưa kịp sửa có gói kèm sản phẩm đưa ra? Các lỗi không quan trọng có gây ra vấn đề gì không? Khách hàng có vì thế mà gặp khó khăn khi thao tác? Do trong giai đoạn đưa sản phẩm ra, ổn định là điều phải suy nghĩ đầu tiên, nên trước khi quyết định sửa một lỗi, cần phải xem có thể vì thế

mà gây ra nhiều lỗi khác không; Có nhiều lỗi được sửa qua loa rất nguy hiểm, lỗi bản thân không có nguy hại lớn, liệt kê chúng vào *tệp tin hướng dẫn* (readme.txt hay readme.doc) để nói rõ sẽ tốt hơn.

Có lúc, cũng như trận đau kéo dài của một người vậy, công việc đưa sản phẩm ra mà không thuận lợi, có thể kéo dài thời hạn tung ta thị trường, mấy nguyên tắc sau dùng để lãnh đạo nhóm phát triển tham khảo để tránh tình trạng bị kéo dài khi đưa sản phẩm ra.

NGUYÊN TẮC 48: QUAN TÂM NHIỀU HƠN YÊU CẦU.

Violate at least one sacred cow

Là người lãnh đạo nhóm, có thể bạn sẽ ép cấp dưới lao vào làm việc. Nhưng thực tế, thách thức phải đối mặt trong giai đoạn đưa sản phẩm ra có thể thự sự cần mọi người lao vào làm, bạn không thể "ra lệnh" mọi người hy sinh tất cả cho công việc. Tôi đã gặp một số nhà quản lý không có nhân tính và cũng không hiểu nhân tính, yêu cầu thành viên làm thêm buổi tối, cuối tuần, kết quả thành viên nếu không coi nhẹ mệnh lệnh thì cũng chê cười vị quản lý này.

Trong thực tế của giai đoạn dưa sản phẩm ra, tác dụng của quản lý vẫn chưa hết, có một số động tác có tính nghi thức, tuy rất nhỏ và đơn giản nhưng lại có ý nghĩa rất lớn. Lúc này, lãnh dạo nên có một số hành động thực tế để biểu đạt quan tâm tới các thành viên, khẳng định cống hiến của họ với tổ chức, cho mỗi thành viên đều cảm nhận được vai trò quan trọng của mình, sự quan tâm của lãnh đạo có thể tăng lực hướng tâm của thành viên, có ảnh hưởng rất lớn đối với việc đưa sản phẩm thuận lợi ra thị trường.

Hoà hợp giữa lãnh đạo và thành viên cần có tính khích lệ, không phải ép làm thêm mà là cung cấp môi trường làm thêm thuận lợi. Ví dụ, viết trên tấm card nhỏ đưa cho một thành viên nào đó (Không dùng từ khách sáo giả dối hay công thức, cần quan tâm thật lòng), vài câu khen ngợi khi tình cờ gặp, dịch vụ chăm sóc trẻ khi làm thêm, tự rút túi trả tiền ăn tối, hoặc tạo điều kiện cho thành viên làm việc tại nhà (Các thiết bị liên lạc đơn giản nối với văn phòng); Những động tác này có chi phí thấp, nhưng lại cho thành viên biết được sự quan tâm và thành ý của bạn mà cảm thấy sự nỗ lực của mình là xứng đáng. Hơn nữa, những động tác như trên thường không theo các nguyên tắc truyền thống, vì họ mà bạn phá vỡ các nguyên tắc truyền thống, cho thành viên hiểu rằng sự khẳng định của bạn đối với họ vượt quá cả quy định mà công ty cho phép bạn. Thông thường, nhân viên phát triển xuất sắc ít nhiều đều có tâm lý chống đối, thích thách thức với uy quyền truyền thống, bạn dùng phương pháp đặc biệt cổ vũ họ sẽ hiệu quả hơn nhiều so với các phương pháp truyền thống.

Các nhân viên phát triển xuất sắc ít nhiều có tâm lý thách thức truyền thống.

Dùng hành động biểu thị sự quan tâm, đồng thời cho các thành viên biết đưa sản phẩm ra là một việc lớn đúng để kỷ niệm. Mọi người chụp một bức ảnh, viết một câu chuyện, thu thập một số truyện cười trong quá trình phát triển, gửi email chúc mừng mọi người cuối cùng sản phẩm cũng ra đời, hoặc biểu dương các thành viên hay sự việc kiệt xuất. Đây là những việc mà lãnh đạo có thể làm vào lúc này, để mọi người lưu lại ấn tượng khó quên.

Hãy hưởng thụ kết quả vui mừng này! Vì mọi người đã vất vả gieo cấy. Sản phẩm ra đời là mục đích nhóm đã hình thành, và cũng là điểm cao nhất trong sự đồng cam cộng khổ của mọi người. Lúc này ý chí thành viên lên đến đỉnh điểm, những hành vi ấu trĩ hay tranh chấp trước đây đều lùi vào dĩ vãng. Mọi người đã hoàn thành nhiệm vụ, sản phẩm ra đời rồi.

NGUYÊN TẮC 49: BẢN THỬ NGHIỆM BETA KHÔNG PHẢI LÀ LÚC SỬA ĐỔI CHỨC NĂNG

Beta is not the time to change

Hầu hết mọi người đều hiểu lầm rằng bản Beta chính là mời các giới bên ngoài đưa ra kiến nghị sửa đổi phương diện thiết kế đối với sản phẩm, sau đó công ty phần mềm tăng cường hay sửa đổi chức năng. Điều này hoàn toàn sai lầm. (*Thử nghiệm Alpha là dùng thử nội bộ, thử nghiệm Beta là dùng thử bên ngoài, bản Beta là đưa sản phẩm sắp xuất bản cho một số khách hàng dùng thử, mặt khác còn gọi là thăm dò thị trường*). Mục đích thử nghiệm Beta là xác định sản phẩm có chạy bình thường trong các hệ điều hành và phần cứng dự kiến không. Tuy ý kiến phản hồi của bản Beta rất có giá trị tham khảo, nhưng trừ phi trong bản Beta phát hiện các vấn đề lớn của sản phẩm, nếu không không nên sửa đổi chức năng, nhiều lắm cũng là sửa lỗi thôi. Các kiến nghị và phản hồi đều giữ lại để suy nghĩ đưa vào trong phiên bản tới.

Làm như vậy không phải là coi thường ý kiến khách hàng, mà ngược lại, nếu bạn muốn đưa ý kiến về bản Beta vào, bạn sẽ không thể nào đưa sản phẩm ra. Quan hệ giữa bạn và khách hàng phải càng thân mật và lâu dài (Xem quan hệ khách hàng trong phần 1).

NGUYÊN TẮC 50: BẢN BETA LÀ ĐỘNG TÁC KHỞI ĐỘNG

The Beta is for spin development

Ấn tượng đầu tiên của khách hàng đối với sản phẩm là thường quyết định đánh giá của họ với sản phẩm. Về cơ bản, phản ứng của người tiêu dùng Beta cũng là phản ứng của đa số khách hàng. Nhân viên kinh doanh dựa vào cơ hội thử nghiệm Beta để hiểu được cảm nhận của những người thử nghiệm với sản phẩm, phân loại và ghi chép các phản ứng để nắm được hướng đóng gói , truyền đạt thông tin sản phẩm. Dù lúc này các thông tin cơ bản đã công bố cho giới truyền thông, nhân viên kinh doanh nhạy cảm vẫn biết được các thông tin nên được giảm đi hay tăng lên từ trong kết quả thử nghiệm Beta. Nhân viên kinh doanh cần xây dựng mô hình tâm lý học khách hàng để có thể hiểu được trạng thái tâm lý và thay đổi khi sử dụng sản phẩm, đồng thời phải điều chỉnh thông tin sản phẩm dựa vào phản ứng tâm lý của khách hàng như thế nào.

Không nghi ngờ gì, bạn nhất định sẽ nhận được đánh giá tiêu cực từ bản Beta, điều này rất bình thường, mà còn nên nằm trong dự liệu của bạn. Nếu

phản ứng tiêu cực rất mạnh, bạn phải phải tìm ra các ưu điểm khác của sản phẩm để cân bằng; nếu phản ứng tiêu cực rất phổ biến, người dùng nào cũng có, bạn phải giảm bớt kỳ vọng của mọi người vào sản phẩm, để phản ứng tiêu cực này nằm trong dự liệu của khách, không để họ có cảm giác bị lừa, phản ứng tiêu cực sẽ giảm thiểu; sau cùng, bạn tìm cách để cho các khuyết điểm của sản phẩm không đến nỗi nghiêm trọng như mọi người phản ánh, chẳng hạn đưa ra phương thức giải quyết, ...

Cứ coi như hoạt động thử nghiệm Beta không do nhân viên kinh doanh dẫn đầu, thì cũng nên để họ tham gia một cách chặt chẽ. Lúc này hoà hợp với mọi người bên ngoài là mấu chốt thành bại của sản phẩm, bây giờ không phải lúc dùng hoạt động phát triển để xây dựng hình tượng sản phẩm.

NGUYÊN TẮC 51: THUẬT CẤP CỨU

Triage ruthlessly

Nếu phần mềm đại diện nhóm phát triển, vậy nói thực lòng, nó có rất nhiều khuyết điểm. Khiếm khuyết là tính chất tất nhiên của mọi tài sản trí tuệ. Mọi việc trên thế giới đều không hoàn mỹ, do đó vấn đề không ở chỗ phán đoán sản phẩm tốt hay xấu, mà là quyết định nên sửa chỗ nào để nó có thể được khách hàng tiếp nhận hay yêu thích. Chúng tôi gọi quá trình phán đoán và sửa đổi này là "thuật cấp cứu".

Thuật cấp cứu, đương nhiên là danh từ y học, trong phòng cấp cứu bác sỹ phải nhanh chóng kiểm tra mọi vấn đề, sử dụng biện pháp cấp cứu cần thiết, sau đó chữa trị theo cấp độ ưu tiên.

Thuật cấp cứu phần mềm cũng có khái niệm tương tự, trước hết phân tích lỗi và sai sót và tính nghiêm trọng của chúng, dưới đây là các nguyên tắc phán đoán có phải chữa trị không?

- Tính nghiêm trọng của lỗi: Nếu lỗi nghiêm trọng đến mức phải quay lại lúc đưa sản phẩm ra thị trường thì phải sửa ngay lỗi này.

- Mức độ rõ rệt: Lỗi có được người dùng phát hiện ngay lập tức và rất rõ rệt không? Có ảnh hưởng đến chất lượng sản phẩm? Có ảnh hưởng đến hình tượng sản phẩm mà trở thành thứ để đối thủ công kích hay trêu cười?

- Phạm vi ảnh hưởng: Tỷ lệ thời gian sử dụng được bao lâu thì gặp lỗi này? Có phải hầu hết người dùng đều gặp phải không?

- Rủi ro khi sửa lỗi: Nếu muốn sửa lỗi này, có gây ra tính không ổn định cho phần mềm? Điều này cần nhân viên phát triển cao cấp và nắm vững phần mềm phán đoán.

- Động thái nhóm: Để sửa lỗi này có lần nhiều thành viên không, hay chỉ cần vài người là được? Có làm cho nhân viên phát triển vốn đang bận bù đầu phải gánh thêm trách nhiệm?

- Giá thành thử nghiệm sau khi sửa lỗi: bất cứ bộ phận nào sau khi sửa lỗi đều cần thử nghiệm, nhóm có đủ người để làm công việc thử nghiệm? Thời gian có cho phép thử nghiệm không?

Thông thường, nhiệm vụ của nhóm nhỏ cấp cứu là chọn và sửa lỗi quan trọng nhất của sản phẩm, cố gắng làm cho hầu hết khách hàng đều vui vẻ trong hầu hết thời gian sử dụng. Đối với người dùng sản phẩm không

hoàn thiện này là cứu cánh hay tai hoạ? Dùng sản phẩm không hoàn thiện này hay không dùng thì tốt hơn? Vấn đề phức tạp và nghiêm túc này, đòi hỏi nhân viên phát triển không ngừng suy nghĩ tình huống của người sử dụng, thăm dò tâm tư của họ, hoà hợp với họ mới có thể tìm ra đáp án chính xác.

NGUYÊN TẮC 52: ĐẢM BẢO TÍNH ỔN ĐỊNH CỦA PHẦN MỀM

Don't shake the Jell-O

Trong giai đoạn hoàn thành, bạn phải truyền cho các thành viên quan niệm: sửa phần mềm là việc rất nguy hiểm. Phần mềm chuẩn bị ra đời, lúc này ổn định là nhiệm vụ quan trọng nhất. Giá thành sửa lỗi và rủi ro mà quá cao thì tốt nhất đừng sửa đổi; những sửa đổi không cần thiết thì phải tuyệt đối tránh.

Sản phẩm ra mắt cũng như một quả lắc, khi bạn lắc nó sẽ chuyển động và ngừng lại. Cũng như vậy, bạn sửa bất cứ lỗi nào đều không tránh khỏi ảnh hưởng đến tính ổn định của toàn bộ phần mềm, đợi khi phần mềm của bạn có thể ngừng lại như quả lắc, chúc mừng bạn, đã đến lúc tung sản phẩm ra thị trường rồi!

Sau đó, nó lại tiếp tục rung lắc!

PART

PHẦN 4
GIAI ĐOẠN CÔNG BỐ

Mục đích chính của cuốn sách tuy là mô tả cách phát triển phần mềm lớn dựa vào phương thức nhóm, nhưng dù sao cũng không thể bỏ qua việc hoà hợp với bên ngoài. Phần mềm mà không thể làm cho những người thông thường nhận ra ưu điểm vĩ đại, độc đáo của nó thì còn có tác dụng gì chứ? Theo tôi, phần mềm vĩ đại cần phải trải qua quá trình hoà hợp với mọi người, làm cho ai cũng nhận biết nó, đã là phần mềm lớn thì phải để nó lưu danh trong lịch sử. Do vậy, sau khi sản phẩm phần mềm hoàn thiện, cần có cuộc họp công bố sản phẩm.

Cuộc họp công bố sản phẩm là thời điểm đưa sản phẩm lưu danh thiên hạ. Khách hàng, phóng viên và các nhà bình luận đều nhìn thấy tác phẩm phần mềm của bạn, là điểm mở đầu để họ tìm hiểu về phần mềm. Dù sản phẩm của bạn không liên qua đến thương mại, cũng không đóng góp đặc biệt nhưng vẫn cần cuộc họp giới thiệu sản phẩm để nhấn mạnh đến tính kỹ thuật của nó, tối thiểu cũng phải giới thiệu kỹ thuật mới tại các cuộc hội thảo khoa học, nơi tập trung nhiều nhân vật quan trọng. Dù dùng phương thức nào hay nhấn mạnh trọng điểm gì, mục đích của bạn là cho mọi người biết một sản phẩm mới ra đời, mong mọi người có nhận thức đúng đắn và rõ ràng về nó. Đây là lần đầu tiên sản phẩm xuất hiện, phải làm cho nó thu hút ánh mắt mọi người, cố gắng để thông tin của nó truyền đến mọi

người một cách rõ ràng, đây là cơ hội xây dựng quan hệ giữa nhóm phát triển và khách hàng, đồng thời là khích lệ lớn cho thành viên.

Giới thiệu sản phẩm là một hoạt động đầy không khí thành công.

Thông tin phát ra không thể thu hồi lại. Tuy có thể phát lại thông tin, nhưng kết quả đều không lý tưởng, ấn tượng đối với sản phẩm của mọi người không phải là thông tin mới, mà là một đám hỗn loạn, do đó lần phát thông tin đầu tiên rất quan trọng. Một cuộc họp giới thiệu sản phẩm được sắp xếp ổn thoả tốt đẹp, việc thiết kế thông tin và không khí đều phải lấy sản phẩm làm trung tâm, mọi sắp xếp đều nhằm thu hút sự chú ý của mọi người, đảm bảo sản phẩm xuất hiện trong điều kiện thông tin chính xác, kết hợp hài hoà với các thiết kế kỹ lưỡng để hoà hợp, giúp mọi người có nhận thức mới và chính xác về sản phẩm. Điểm tốt khi làm thành công cuộc họp giới thiệu sản phẩm có rất nhiều, dưới đây chỉ khái quát vài điểm:

- Cuộc họp giới thiệu xác định, sẽ giúp nhóm cùng hợp lực vì mục tiêu xác định. Nếu mỗi người cho rằng/kỳ vọng một việc gì đó xảy ra, cũng như một

mục tiêu chung, vô hình chung thúc đẩy nhóm coi đó như một thách thức đầy ý nghĩa mà đồng tâm hiệp lực. Tôi đề nghị mỗi nhóm tham gia hoạt động giới thiệu sản phẩm đều có chiến lược chi tiết, vì bản thân cuộc họp và sản phẩm đã như một trận chiến. Nhóm giới thiệu sản phẩm cũng như nhóm phát triển, áp dụng nguyên tắc 54, cần có mục tiêu chung, biết cách vận dụng các sách lược, có tinh thần đồng đội; nhóm giới thiệu cần coi nhiệm vụ như sự giao ước toàn tâm toàn ý. Toàn thể nhân viên đều cảm nhận được không khí thành công khi giới thiệu sản phẩm, sản phẩm được nuôi dưỡng vất vả, nay đã có trong tay, trong lòng trào dâng một cảm giác dầu phấn chấn, tâm tư này sẽ truyền sang mỗi vị khách trong cuộc họp làm cho cuộc họp giới thiệu sản phẩm càng thêm sôi động.

- Cuộc họp giới thiệu sản phẩm thành công sẽ nâng cao độ tiếp nhận sản phẩm. Có một số ý kiến rất nhạy cảm với cuộc họp giới thiệu sản phẩm. Rất nhiều người tham gia cuộc họp giới thiệu sản phẩm chỉ đơn thuần là yêu thích khoa học kỹ thuật, hoặc sản phẩm có ảnh hưởng lớn đối với bản thân anh ta, hơn nữa bản thân cuộc họp giới thiệu sản phẩm ít nhiều có tính hài hước, do đó sẽ có rất nhiều người đến một cách nhiệt tình, họ hy vọng biết bạn, nhóm của bạn và nỗ lực của bạn, phần mềm của bạn và tình cảm của bạn đối với phần mềm này; bởi vì khi sử dụng phần mềm của

bạn, họ bắt đầu hiểu về bạn, hình tượng về thái độ của bạn cũng có thể là một trong những ấn tượng đối với sản phẩm mới. Trước hết nghiên cứu kỹ với nhân viên kinh doanh, làm thế nào để thể hiện tầm quan trọng của sản phẩm trong lịch sử và kỹ thuật, điều này cũng quan trọng như làm thế nào để làm cuộc họp trở nên vui vẻ.

- Cuộc họp là hoạt động quan trọng nhất đưa sản phẩm ra mắt công chúng. Cuộc họp giới thiệu là sự ra mắt đầu tiên của sản phẩm; công việc sắp xếp cuộc họp và hoạt động sau đó đều có thứ tự lôgic, cần phải được sắp đặt thoả đáng. Trong thử nghiệm Beta, bạn phải mời giới truyền thông, khách hàng quan trọng, nhân vật có tiếng trong lĩnh vực liên quan và bạn bè của bạn dùng thử bản Beta, như vậy họ sẽ có khái niệm rõ ràng về sản phẩm, đồng thời vui vẻ đưa ra cảm nhận của mình, đồng thời làm chứng cho sản phẩm. Và trước ngày giới thiệu, bạn gửi email cho những người dùng thử một đoạn tóm tắt, nhắc họ nhớ tới tính năng và đặc điểm của sản phẩm, để tránh do bận rộn và họ quên mất cảm nhận khi dùng thử. Trong cuộc họp giới thiệu sản phẩm, những người này sẽ nói cho phóng viên cách nhìn nhận của họ đối với sản phẩm một cách tự nhiên, rõ ràng, chẳng khác gì hướng dẫn sử dụng vậy. Sau cuộc họp giới thiệu sản phẩm là hàng loạt hoạt động quảng cáo như tờ rơi, email, catalogue,

trình chiếu,... Bạn phải chú ý chặt chẽ phản ứng của những người tham dự, cố gắng đưa ra hỗ trợ và dịch vụ hoàn thiện. Tất cả các chi tiết này là một phần của cuộc họp, cũng là một phần của thông tin sản phẩm, dù những việc này tốn bao công sức, dù bạn và khác hàng thân thuộc đến mức độ nào thì cũng phải chuẩn bị kỹ các chi tiết này, đảm bảo cuộc họp thành công tốt đẹp nhất.

NGUYÊN TẮC 53: PHẦN MỀM LỚN PHẢI CÓ CÂU CHUYỆN LỚN

Compete with the superior story

Phần mềm lớn, sẽ có một câu chuyện lớn (Sản phẩm bình thường sẽ có câu chuyện bình thường). Kết cấu thông tin đan xen nhau phức tạp như các đơn vị trong phần mềm, bởi thông tin phải truyền cho các đối tượng khác nhau theo các tầng lớp khác nhau, thông tin phải biến thay đổi tuỳ theo về mặt tình cảm với người tiếp nhận, phải diễn biến thay đổi tuỳ theo thời gian và môi trường, thông tin cũng phải đơn giản, như vậy mới gây hứng thú cho người tiếp nhận. Rất rõ ràng, chất lượng hoà hợp do thông tin truyền đạt có quan hệ rất lớn đến thành công của kinh doanh sản phẩm. Hãy ghi nhớ, bạn phải đầu tư thích đáng cho truyền đạt thông tin, quan trọng như công việc phát triển vậy, bởi vì thông tin và sản phẩm liên quan chặt chẽ với nhau, cả hai đều là tài sản trí tuệ do nhóm tạo ra.

Trí tuệ truyền thống qua nhiều năm truyền tụng, rất dễ được mọi người tiếp nhận.

Trước hết bạn phải suy nghĩ kỹ, xây dựng kết cấu thông tin dựa vào chiến lược hoà hợp sản phẩm cần phải như thế nào? Làm thế nào để họ hiểu cách sử dụng sản phẩm? Thông tin của bạn phải bao gồm những điều này mới có thể làm cho tài sản trí tuệ vô hình được mọi người nhìn thấy.

Khi đứng trước mặt mọi người, không chỉ chú ý đến tính năng sản phẩm mà trước hết hãy nói qua về nội hàm của sản phẩm, kể câu chuyện của sản phẩm cho nhóm, phóng viên, các nhà phân tích, khách hàng và những người đầu tư nghe. câu chuyện này phải có màu sắc trí tuệ truyền thống. Đã là những thứ mang màu sắc trí tuệ truyền thống đều sẽ được truyền tụng một cách tự nhiên, đồng thời là ngôn ngữ chung trong bất cứ lĩnh vực nào.

Trí tuệ truyền thống phải là câu chuyện nhỏ đạo lý lớn. Câu chuyện của bạn tối thiểu phải có một trục liên quan đến xu hướng khoa học kỹ thuật hay tình hình thị trường (hoặc cả hai), chỉ như vậy người nghe mới cảm thấy câu chuyện có giá trị, vui vẻ lắng nghe. Nội dung câu chuyện phải đơn giản, đồng thời lại chỉ dẫn suy nghĩ sâu sắc mọi người, cho những người có đầu óc linh hoạt cảm thấy có một số nội hàm mà mình đã nghĩ tới (sự cộng hưởng!); những lời nói quá huyền bí không thể làm các chuyên gia hứng thú, đồng thời

không thể trở nên bình dân. Cần phải chú ý câu chuyện vừa đơn giản vừa có nội hàm, không được đơn giản quá mà thành vô nghĩa, nội hàm phải làm cho mọi người cảm thấy rất có giá trị mới được.

Phải làm thính giả đồng cảm: "Đúng vậy! Trước đây tôi đã từng nghĩ thế".

Muốn kể một câu chuyện đơn giản về thông tin sản phẩm thì quả là một điều rất khó. Bạn phải làm rung động trái tim thính giả, để họ đồng cảm "Đúng! Trước đây tôi đã từng nghĩ thế". Đây không phải tuyên truyền quan niệm của bạn, mà là giao lưu tư tưởng với thính giả một cách vô tình, dẫn dắt họ suy nghĩ vấn đề mfa bạn đưa ra, để thính giả dùng tâm của mình lãnh hội thông tin của bạn, chứ không phải bị bạn "rót" thông tin. Các thông tin phải nghe như kiến thức thông thường, bạn nói ra và cộng hưởng trong lòng họ, như vậy thính giả sẽ nghĩ đó là chân lý.

Chúng ta hãy lấy một ví dụ nói rõ nội hàm của thông tin, bạn có thể mở màn như thế này: "Người sử dụng không thích dùng footbar vì nó rất khó sử dụng". Như vậy chẳng khác gì nói thẳng một cách trái với truyền thống - "hoàng đế cởi truồng". Dùng một sự thật mà mọi người đều biết hoặc muốn tin (có lẽ mọi người

ngại nói ra), từ sự thật này mà phát triển sang sự thật khác liên quan đến sản phẩm (Bạn phải biểu đạt trọng điểm này, có lẽ nói trực tiếp sẽ không được mọi người tiếp nhận), đồng thời không được kể lại sự thực mà người khác đã nhắc đến (Mọi người sẽ lầm lẫn không biết câu chuyện này là do ai kể).

Khi chúng tôi giới thiệu Visual C^{++} 1.0, sự thực mà chúng tôi nhấn mạnh là đa số các lập trình viên vẫn chưa chuyển từ C sang C^{++}. Dữ liệu từ kết quả điều tra của chúng tôi cho thấy, có 80% lập trình viên ngôn ngữ C dự định, chuẩn bị hoặc mong muốn chuyển từ C sang C^{++}, nhưng chỉ có 10% người thực sự làm được. Các báo cáo phân tích của ngành và giới truyền thông đều biểu thị thời đại C^{++} đã đến gần, nhưng tư liệu của chúng tôi lại cho tình hình ngược lại, đó chính là điều mà chúng tôi nói thực - bộ quần áo mới của hoàng đế

Trực tiếp nói ra thông tin sản phẩm, chưa chắc đã là phương thức biểu đạt hữu hiệu.

Việc nói thật của chúng tôi, nhanh chóng làm rung động trái tim của mọi người, bởi đó là sự thật không thể chối cãi, hơn nữa đa số mọi người đều có đồng cảm. Cứ coi như chúng tôi không nói, cuối cùng ai cũng sẽ biết chân tướng sự thật, chỉ cần mất thêm chút thời gian và hoài nghi nữa.

Sau đó, bước thứ hai là phát triển sang sự thật khác, tức là thông tin sản phẩm, rất đơn giản: dù là lập trình viên thành thạo C thì C^{++} cũng rất khó học - cho đến khi Visual C^{++} xuất hiện mới thôi; chúng tôi không truy tìm tại sao C^{++} đã làm cho nó trở nên đơn giản.

Trí tuệ truyền thống thường được biểu đạt qua ngạn ngữ, dùng một hai câu đơn giản nhưng nổi trội, để mọi người dễ thuộc, dễ nhớ khó quên. Chính trị gia đều là những cao thủ về sáng tạo ra khẩu hiệu, chẳng hạn: "Ngoài lo sợ bản thân, chẳng còn gì đáng sợ nữa", "Đừng hỏi tổ quốc đã làm gì cho ta, mà hãy hỏi ta đã làm gì cho tổ quốc...", ... Những khẩu hiệu này sở dĩ dễ đi vào lòng người là do chúng có trí tuệ truyền thống, đơn giản dễ hiểu, lại có chức năng dẫn dắt, không cần các lý luận dài dòng. Tuy người đưa ra khẩu hiệu đã ra đi từ lâu, nhưng đến nay chúng vẫn tồn tại, mà còn trở thành ngọn đèn sáng cho trí tuệ, vì khi mọi người mê muội chúng đưa ra chỉ dẫn, nếu những khẩu hiệu này không có nội hàm trí tuệ, sau khi thịnh hành sẽ tự tiêu tan, không ai còn nhớ.

Câu chuyện hay sẽ làm mọi người nhớ mãi, bởi nó có nội hàm, có kiến giải, cho người nghe cảm nhận được trí tuệ bên trong. Câu chuyện hay sẽ làm mọi người dễ liên tưởng, nhất là khi gây ra cộng hưởng nội tâm (Vì đó là sự thực đơn giản), mọi người sẽ mở rộng lòng để

đón nhận câu chuyện liên tưởng, phát triển phía sau, mức độ tiếp nhận tỷ lệ thuận với mức độ làm họ rung động. Chẳng hạn, "Đội bóng này luôn không thể giành chiến thắng trong các trận đấu quan trọng", đây là sự thật thứ nhất, nhưng sau khi nghe xong trong lòng mọi người tự nghĩ: "Đội bóng này không giỏi" hay "Nếu không thua trận quan trọng thì sẽ là đội bóng khá tốt", đó là phản ứng tất nhiên của mọi người, đưa quan niệm của mình vào thông tin tiếp nhận được, dùng lời của mình để diễn tả lại.

Bạn hướng mọi người tự nghĩ ra sự việc, chắc chắn còn có sức thuyết phục hơn là bạn nói ra.

Do đó, nếu bạn có thể kể câu chuyện sản phẩm dựa vào phương thức các tình tiết thì thật tuyệt: có bối cảnh, có không khí, tình tiết thay đổi, có người tốt cũng có kẻ xấu, ... Tìm một chuyên gia kể chuyện hỗ trợ bạn kể câu chuyện sản phẩm sinh động hơn, lại làm rung động nội tâm thính giả, đồng thời dạy cho các thành viên đều có thể kể câu chuyện này.

Một câu chuyện hay phải dựa vào một sự thực dễ thấy. Nhưng không được có quá nhiều sự thực, một lỗi hay phạm phải của nhiều người là cung cấp quá nhiều thông tin, kết quả làm rối loạn, thiếu trọng điểm, ấn

tượng của mỗi người một khác, như vậy sẽ rất khó hoà hợp. Sự thực mấu chốt không được vượt quá ba, đưa ra sự thực mà họ ghi nhớ được, vừa đơn giản vừa quan trọng, đồng thời phải xác định sự thực quan trọng của bạn liên quan chặt chẽ đến câu chuyện sản phẩm.

Câu chuyện phải làm cho nhiều nhân vật quan trọng nói đi nói lại, tạo ra ấn tượng cực kỳ sâu sắc, thậm chí để họ tự kể lại cho bạn bè mình nghe. Do đó, câu chuyện phải dựa vào vài sự thật quan trọng. Lấy trọng điểm là nhân tính sẽ làm cho người nghe không bao giờ quên.

Đáng ngạc nhiên, phương pháp tốt nhất là không nên đưa ra vấn đề quan trọng mà hãy để cho người nghe tự đưa ra kết luận chính xác. Ví dụ, khi giới thiệu Visual C++, điều quan trọng nhất là cho giới truyền thông, các nhà phân tích và mọi người hiểu: Microsoft muốn quay lại thị trường C++, với quyết tâm giành vị trí thống soái, chúng tôi quyết tâm thay đổi càn khôn. Nhưng trong bất kỳ thông tin nào về kinh doanh đều không nhắc tới chuyện này, thậm chí cũng không đề cập trong câu chuyện sản phẩm. Đạo lý này rất đơn giản, nếu chúng tôi khoe khoang về sự vĩ đại của sản phẩm khắp nơi, vừa không làm mọi người tin phục vừa tạo ra ác cảm. Ngược lại, chúng tôi để thông tin này

được suy đoán ra từ trong nội tâm thính giả, trong câu chuyện sản phẩm đưa ra các manh mối để mọi người tự kết luận. Câu chuyện mà bạn dẫn dắt thính giả nghĩ ra chắc chắn sẽ có sức thuyết phục hơn so với bạn kể cho họ.

NGUYÊN TẮC 54: XÂY DỰNG HÌNH TƯỢNG CHIẾN THẮNG

Create a winning image

Ngoài việc làm được sản phẩm tốt, củng cố địa vị thị trường, do một vài lý do sau mà bạn còn cần xây dựng hình tượng chiến thắng.

- Khách hàng cần có cảm giác tương đồng: phần mềm cũng như các hàng hoá khác, khách hàng mong muốn có cảm giác tương đồng đối với sản phẩm mình mua. Do khách hàng bỏ nhiều thời gian sử dụng nên họ mong được coi là "chuyên gia phần mềm". Từ đó có thể thấy, khách hàng không hề "mua" phần mềm mà là "lựa chọn" phần mềm, điều này đáng để cho những người trong ngành phần mềm chú ý.

- Khách hàng mong muốn rằng những gì mình có là những thứ mà người khác khát vọng hoặc ngưỡng mộ. Phần mềm tượng trưng địa vị thân phận khách hàng hoặc "đồng nghiệp" của họ. Trong thời đại khoa học kỹ thuật máy tính không ngừng thay đổi, những thứ mới nhất (dù phần mềm hay phần cứng) có nghĩa là thịnh hành nhất, cao cấp nhất, mong muốn mới nhất tốt nhất là một loại động cơ rất mạnh. Nếu bạn cho rằng mình không bị ảnh hưởng bởi những gì

thịnh hành, hãy xem khi bạn chọn mua quần áo - cà vạt - dày dép, bạn sẽ thấy trong lòng nổi dậy ham muốn thì lúc đó bạn sẽ hiểu rằng khó có thể cưỡng lại những gì đang thịnh hành (Tham khảo nguyên tắc 14).

- Khách hàng biết tầm quan trọng hỗ trợ của hãng, không muốn mình là người dùng đơn độc. Khách hàng không đơn thuần mua phần mềm mà là xây dựng mối quan hệ lâu dài với bạn, họ biết rằng sau khi quyết định sử dụng phần mềm của bạn, sẽ phải dựa vào tiến bộ của các phiên bản mới trong một thời gian dài, họ phải đầu tư rất nhiều thời gian tiền bạc để học cách sử dụng phần mềm, họ không muốn sau khi mua phần mềm thì hãng sập tiệm, mình trở thành đơn độc, dùng cũng không được mà không dùng cũng không xong.

Hoà hợp với bên ngoài không chỉ là thông tin sản phẩm, mà còn là hình tượng người chiến thắng! ồ, ... đừng quên nguyên tắc 53, không dùng phương thức nói trực tiếp, người chiến thắng không tuyên truyền sự chiến thắng của mình ở khắp nơi, càng không lộ ra mong muốn chiến thắng, như vậy lần nào anh ta cũng chiến thắng.

LỜI KẾT

Nếu nhất định phải dùng một câu hỏi để tổng kết, tôi sẽ nói: phát triển phần mềm tạo nên nhân viên phát triển.

Có lẽ trong tình tiết của bộ phim "Chinatown", Roman Polanski đã đưa ra kết luận về phát triển phần mềm: Nhân vật chính trải qua giằng xé nội tâm khá lớn, gần như gục ngã, bạn anh ta vẫn dùng giọng trêu đùa để gây cười: "Hãy cười lên, đó là Chinatown".

Đương nhiên, phát triển phần mềm không có câu kết thúc, bạn phải bắt đầu từ đầu cuốn sách, sau đó từng mục từng mục cứ như vậy luân hồi,...

Các bạn, hãy cười lên, đó chính là phát triển phần mềm!

PHỤ LỤC

GIỎI DÙNG NGƯỜI

Đối với sáng tạo tài sản trí tuệ, quan trọng nhất là bộ óc thông minh của con người, do đó phụ trách dự án rất mong có được nhân viên thông minh, chăm sóc họ cẩn thận để họ làm ra những phần mềm thông minh, đồng thời có thể xây dựng được nhóm phát triển xuất sắc, không ngừng trưởng thành và sáng tạo, luôn đưa đến cho khách hàng những thứ mới nhất tốt nhất.

Nghe có vẻ như những điều các cụ già hay nói, nhưng thực hiện được cần có học vấn rất rộng.

THUÊ NHÂN VIÊN THÔNG MINH

Muốn tìm người thông minh không phải dễ, nhưng cố gắng trong việc này cũng đáng, bởi người tài giỏi có thể sinh tồn. Trong ngành phần mềm, một người phải trải qua nhiều kinh nghiệm tiêu cực, như ông chủ không có tầm nhìn, đưa ra quyết sách sai, mất thời gian làm những phần mềm vô dụng, làm tan vỡ nhóm, tiến độ tụt hậu nghiêm trọng. Những điều này là thách thức dau đớn đối với bất cứ một lập trình viên nào, nhưng đây lại là hiện tượng bình thường trong ngành phần mềm; "Thông minh" là điều kiện có lợi nhất, dựa vào thông mình, họ có thể không quan tâm đến những khó khăn này, thậm chí giải quyết được rủi ro của nhóm, đem lại nguồn thành công. Đương nhiên, nguyên liệu để sáng tạo tài sản trí tuệ là thông minh tài trí, do đó bạn phải tìm được người thông minh nhất và cho vào nhóm của bạn.

Khi bạn phỏng vấn một người thông minh, cần chú ý đến mấy loại đặc trưng nhân cách sau:

1. Chú ý tượng trưng tài trí không biểu hiện qua ngôn ngữ

Hướng phát triển tài trí mỗi người một khác, có lúc người thông minh lại chẳng ra sao, bạn phải chú ý đặc tính riêng của mỗi người, không nên chỉ câu nệ vào

các ấn tượng "thông minh", chú ý quan sát thông minh tự nhiên, tuy có thể là một loại hình thái mà bạn không dự liệu.

Quan sát kỹ các phản ứng của người dự tuyển, anh ta biểu hiện cá tính như thế nào? Điều đó lộ ra thông tin gì? Từ biểu lộ nét mặt xem thay đổi nội tâm, phân tích phán đoán xem tiềm lực của anh ta. Luyện tập quan sát người, bạn sẽ phát hiện ra rất nhiều thông tin ẩn giấu, so với biểu hiện bên ngoài bạn sẽ dễ phán đoán chính xác hơn về người trước mặt.

Dùng phản ứng tình cảm và trí tuệ của bạn để xem xét anh ta, người dự tuyển có đối phó với thức thức tốt hơn bạn, hoặc đánh giá xu thế nhanh hơn, yêu cầu của anh ta có vượt khả năng đáp ứng của bạn? Cảm giác phỏng vấn như thế nào? Anh ta có chỗ đáng để bạn học hỏi không?

Thời gian phỏng vấn không dài, vai trò của hai bên là cố định. Phản ứng của người dự tuyển có như bạn dự đoán? Có hành vi lời nói nào làm bạn bất ngờ? Có chủ động phá vỡ vai trò người xin việc truyền thống một cách tự nhiên và thay đổi không khí trò chuyện? Nếu không, bạn có thể phá vỡ ranh giới cố định, làm anh ta bỏ đi hình tượng cứng nhắc của người xin việc? Anh ta có đủ sức bật, không khư khư giữ ý kiến của mình? Có bộc lộ ham muốn học hỏi, hay chỉ vì sợ bị đào

thải nên mới tìm kiếm kiến thức mới? Nếu người dự tuyển đủ thông minh, anh ta sẽ làm cho quá trình phỏng vấn có một số thay đổi, nếu anh ta có năng lực ảnh hưởng đến quá trình phỏng vấn, anh ta cũng có thể ảnh hưởng nhóm phát triển hay những sự việc phức tạp khác.

2. Dùng vấn đề khó hỏi người ứng cử

Bạn dưa ra một vấn đề thực tế, cung cấp các thông tin bối cảnh có liên quan, đề nghị anh đưa ra ý kiến. Đương nhiên người ứng cử biết mình đang được kiểm tra, do đó có thể bạn sẽ nhận được đáp án chuẩn xác đã được chuẩn bị trước, nhưng quan trọng không phải đáp án mà là quá trình suy đoán của anh ta, từ đó biết được thái độ làm việc và suy nghĩ vấn đề.

Người ứng cử có nhanh chóng đưa ra giả thiết? Hầu như ai cũng như vậy. Nếu không, người ứng cử sẽ phải yêu cầu có thêm nhiều thông tin, từ đó định ra kết cấu vấn đề, từ đó thử đưa ra đáp án. Có lúc tôi cố tình đưa ra một vấn đề khó có thể tìm ra đáp án nhưng có vẻ tôi xác định có đáp án, xem phản ứng của người ứng cử. Người thông minh sẽ không rơi vào bẫy tự biện bạch vì đáp án của anh ta thế nào cũng sai, mà ngược lại sẽ hỏi tôi, cuối cùng phải như thế nào? Đương nhiên, đa số mọi người cố chấp và có tâm lý phòng vệ, và không hiểu rằng xin giúp đỡ mới là biện pháp tốt.

Thử xem nếu bạn chủ động giúp đỡ, phản ứng của anh ta sẽ ra sao, sẽ có rất nhiều kết quả khác nhau, tôi thích nhất là trước hết hỏi tiếp nhận giúp đỡ có phải là biểu hiện yếu đuối không, sau đó mới quyết định có nhận giúp đỡ không.

3. Thử dạy họ

Nói về lĩnh vực chuyên môn của họ bạn xem phản ứng của người ứng cử, anh ta có nhanh chóng tiếp thu những gì bạn nói không? Có hỏi về các vấn đề sâu hơn? Tiếp nhận hay phản đối? Có thể đưa ra biện pháp hay hơn không?

Hãy ghi nhớ mục tiêu của bạn là tuyển người đảm nhận trọng trách lớn, sau này sẽ trao dủ quyền cho anh ta, anh ta sẽ thay bạn quản lý rất nhiều kỹ thuật, thậm chí quyết định một phần tương lai của phần mềm. Người ứng cử xây dựng quan hệ như thế nào với hình tượng uy quyền (chính bạn là người đang phỏng vấn), là mấu chốt quan trọng biểu hiện tương lai của anh ta. Người bạn cần, phải dẹp tâm lý chủ quan của mình sang một bênmà chú tâm giải quyết vấn đề hay học hỏi kiến thức mới.

4. Loại bỏ điểm mù trong kiểm tra nhân tài

Tôi đã gặp phải một số sai lầm rất lớn trong tuyển nhân tài, đó là người quản lý quá coi trọng một kỹ

thuật nào đó. Có thể nhìn thấy khuynh hướng này trên quảng cáo tuyển nhân viên: "Tuyển lập trình viên cao cấp, cần biết kỹ thuật lập trình X và Y năm kinh nghiệm". Thực tế, có kỹ thuật là điểm đáng chú ý, nhưng không phải tất cả, quan trọng hơn là những thứ bên ngoài kỹ thuật, dù chúng ta chỉ xem kỹ thuật, kỹ thuật này trong vòng 1 năm có thể bị đào thải, quan sát trọng điểm về kỹ thuật là trong các kỹ thuật liên quan khác, anh ta có thành thạo hay không. Tôi muốn xem quá trình học kỹ thuật và tích luỹ của anh ta: Dùng kỹ thuật gì vào lúc nào? Quá trình học tập đã đủ để chứng minh anh ta có nắm ngay được kỹ thuật chúng tôi đang sử dụng hay không?

PHÂN VIỆC ĐÚNG NGƯỜI

Nếu bạn đã tuyển được một người thông minh, bạn sẽ hy vọng phát huy được khả năng của anh ta, đem lại lợi ích lâu dài cho nhóm. Nếu chỉ có bạn biết tiềm lực của anh ta, mà các thành viên khác lại không thể nhìn thấy, thì những gì anh cống hiến sẽ rất có hạn. Tính cách làm việc của các thành viên có thể chia thành hai loại lớn: dũng cảm nôn nóng và với quá cao. Vào nhóm khác nhau, thời gian khác nhau sẽ có tính cách làm việc khác nhau. Là người quản lý phải biết cấp dưới của bạn thuộc loại tính cách nào, cuộc sống cá nhân của họ ra sao, bạn mới có thể giao cho công việc thích hợp, phân việc đúng người mới phát huy được tài năng.

NGƯỜI DŨNG CẢM NÔN NÓNG THÍCH TIẾT TẤU NHANH

Người dũng cảm nôn nóng có biểu hiện học nhanh, phát triển nhanh, biểu hiện thành công, do đó họ cần không gian rộng rãi mới duy trì đam mê tốc độ của họ, tạo nhiều cơ hội thể hiện, phạm vi công việc gánh vác rộng hơn, hướng dẫn họ đặt mục tiêu vào sự tự phát triển. Nếu người quản lý không nhận ra thiên tài, hoặc giảm tốc độ của họ, họ sẽ chèn ép những người

có thực lực kém, hoặc tự tìm kiếm đất phát huy. Trong một nhóm mạnh không được để tình trạng này.

Cũng như câu danh ngôn mà bố tôi hay nói: "Nếu thức ăn trong máng không đủ, lợn sẽ tự đánh nhau".

Còn con người thì sao, không phải trưởng thành là đứng im bất động. Nếu chu kỳ sản phẩm mới ngày càng dồn dập, tỷ lệ chiếm lĩnh thị trường ngày càng lớn, độ hài lòng của khách hàng cũng mỗi năm một tăng, hay bất kỳ một kết quả có lợi nào, đều biểu thị nhóm của bạn đang phát triển, nếu không sẽ là tụt dốc.

Đối với nhóm, thách thức mà phát triển đem tới là chọn bổ sung nhân tài như thế nào để thích ứng với sự phát triển của nhóm, đồng thời tránh cản trở đó tăng người. Thông thường, vấn đề đầu tiên khi nhóm tăng người là thành viên bắt đầu thấy bất an, niềm vui thành công nhanh chóng bị lãng quên, mọi người bắt đầu nói về những ngày tốt trước đây. Hiện tượng này xuất hiện là do người quản lý không cho thành viên những thách thức vừa phải.

Để thay đổi tâm lý bất an, cần phải kích thích, tìm vài nhân vật quan trọng và hỏi xem họ đang muốn làm gì, cái gì có tính thách thức với họ? Chúng có liên quan gì đến ngành phần mềm? Họ muốn có môi trường như thế nào? Phần mềm sáng tạo trong mơ của họ có

mô hình như thế nào? Sau đó, bạn trao quyền cho họ đi làm những việc thích hợp.

Phát triển, đại diện cho đầu tư mới, bạn có thể làm những thứ hoàn toàn mới, để thành viên đảm nhiệm các vai trò khác với trước đây, không được do dự, ai cũng cần phát triển, từ lãnh đạo cao nhất đến người ở tầng thấp nhất đều không tránh khỏi sự phát triển, kể cả bạn và tôi. Do đó, bạn phải tìm đường ra thích hợp nhất cho sự phát triển của thành viên.

SỨC SÁNG TẠO

Sức sáng tạo bị tâm lý phòng vệ hạn chế, nhưng nếu phòng vệ vừa phải thì rất tốt. Trong một nhóm mạnh, dù sức sáng tạo được mọi người đánh giá cao như thế nào, luôn có một số thay đổi bị chống đối. Cho dù là thay đổi nhỏ đến đâu cũng phải làm mục tiêu tiến bộ, đi sâu hay tăng cường tư tưởng và cách làm vốn có mới có thể được chấp nhận. Do đó, thay đổi phải dựa trên cơ sở vốn có hoặc sự thực đã được mọi người tiếp nhận thì mới thuận lợi. Đáng tiếc là, tuy thay đổi chỉ có hai bước, bước hai được xây dựng trên bước một chứ không phải một sự thật nào đó, đều có thể bị nhóm mạnh nhất từ chối.

Thay đổi thực sự có tính sáng tạo, chỉ được tạo ra trong một môi trường có năng lực tự được phát huy. Trong môi trường này, không chỉ chấp nhận thay đổi

thông thường, coi thay đổi là bình thường, mà còn tạo cơ hội thay đổi, hoan nghênh thay đổi, mượn thay đổi để đưa bản thân vào một môi trường động thái mới. Nhóm tự phát huy coi những thay đổi lớn có tính cách mạng là tư duy đảo ngược, được xuất hiện với bộ mặt khác. Một nhóm mạnh không bằng nhóm có nhiều sức sáng tạo, nhóm có thể rất mạnh nhưng sức sáng tạo hơi không đủ, đây không phải tình huống lý tưởng nhất; nhóm lý tưởng phải tràn đầy sức sáng tạo, luôn có biện pháp mới, dám tìm tòi những lĩnh vực mà chưa ai đặt chân vào, đồng thời có năng lực giành lấy kết quả.

Đáng tiếc, càng là nhóm mạnh lại càng tự hào về các thành tích trong quá khứ, càng không chấp nhận thay đổi có tính cách mạng. Trong bất cứ lĩnh vực nào, đó đều là căn bệnh của các nhóm có biểu hiện chỉnh thể tốt nhất, đáng để chúng ta suy nghĩ.

NGƯỜI VỚI QUÁ CAO MUỐN CÓ NHỮNG ĐỘNG TÁC LỚN

Đôi khi, công việc mà một số thành viên muốn nhận, thực tế vượt quá khả năng của họ. Loại tâm lý "tham nhiều" này có thể bắt nguồn từ kinh nghiệm làm việc trước đây, có lẽ khi đó khát vọng về công việc vẫn chưa được thoả mãn, kết quả là tâm lý trước đây chưa

được thoả mãn chuyển sang công việc hiện tại, tạo ra mong muốn được bay cao, với quá cao: tuy bây giờ là môi trường hoàn toàn có thể phát huy năng lực, ai cũng có điểm mạnh, họ vẫn yêu cầu được trao đủ quyền hợn, làm việc với phạm vi lớn, mong muốn đóng vai trò lớn hơn, luôn không thoả mãn tâm lý thiếu hụt, vì anh ta có ý thức tiềm ẩn tránh việc không làm gì trong nhóm. Việc tham nhiều, chính là mong muốn vượt thực tế và luôn không thể thoả mãn. Do nhóm tự tìm cách để chống lãng phí vô ích, các thành viên khác sẽ không cho họ phát triển quá mức.

Dù là thành viên xuất sắc, khi mới bắt đầu học trưởng thành cũng có tình hình này: có thể anh ta nhận được một công việc với phạm vi nhỏ, nhưng không hiểu rằng phạm vi công việc tuy hơi nhỏ nhưng không ảnh hưởng đến cơ hội biểu hiện bản thân, thế là anh ta trách quản lý phán đoán sai, sử dụng ít tài năng lớn, chôn vùi khả năng của mình. Nếu anh ta biết cách chăm sóc cho cuộc sống của mình, anh ta sẽ có thái độ tích cực đối mặt với việc này, mượn cơ hội này để học tập nhiều hơn. Đáng tiếc, nhiều nhân tài không vượt qua cửa ải này, hoặc khoác áo ra đi, hoặc là quản lý đã cho anh ta quá nhiều công việc không thể tiêu thụ được, kết quả cuối cùng vẫn là bỏ mất cơ hội phát triển một người. Quản lý phải hỗ trợ cấp dưới bình thản qua

giai đoạn này, điều này rất có lợi cho bản thân anh ta cũng như cả nhóm.

Khai thác mục tiêu thật sự

Bạn phải bỏ công khai thác mục tiêu thật sự của những người với quá cao. Họ có một đặc trưng rất rõ rệt, đó là do dự thiếu quyết đoán với mục tiêu quá lớn, mà thường là hai mục tiêu đối lập (cái gì cũng muốn); Điều này rất quan trọng, đó là cơ sở cho công việc của bạn dưới đây.

Đa số các trường hợp, người với cao muốn có hình tượng bản thân, họ muốn lợi dụng nội dung và giá trị công việc để tâng bốc ý nghĩa không thật. Bạn phải thử khai thác, hình tượng mà họ thực sự muốn hay mong biểu hiện gì. Nhưng phương pháp rất đơn giản, hỏi họ vài vấn đề sai là biết ngay: Mục tiêu này có ý nghĩa gì với anh? Anh thấy ý nghĩa này có thích hợp? Anh thấy mình có bao nhiêu chắc chắc đạt được mục tiêu? Nếu anh hoàn thành mục tiêu thuận lợi, anh có muốn thành quả của mình được vận dụng như thế nào? Sau đó kiểm tra cảm giác của bạn: Mục tiêu này mình có thấy thích hợp không? Cách nghĩ của người khác ra sao? Ai là người tốt nhất? Nếu chọn ai đó đảm nhiệm, là vì bản thân hay vì tập thể, hay vì chính bạn.

Xem cách nhìn nhận với phần mềm

Bạn đã biết mục tiêu của họ, bây giờ bạn cần phải quyết định mục tiêu đó có vượt khả năng của họ không, hay không hợp với mục tiêu của nhóm. Tốt nhất là không có vấn đề gì, nếu không bạn phải tìm ra nguyên nhân thực sự. Tại sao anh ta quá tự tin vào năng lực của mình? Đây là hiện tượng ngẫu nhiên hay là thói quen của họ? Bạn phải biết rằng, các thành viên rất nhạy cảm khi có người cho rằng họ "không đủ năng lực, làm không được", bạn phải cho họ thấy rằng bạn luôn đứng ở bên họ, giúp họ đạt được mục tiêu lớn chứ không phải ngăn chặn họ.

Bạn phải hiểu cách nhìn nhận của họ về phát triển phần mềm, nhờ đó để thăm dò lý do thực sự cho những hành vi của họ. Có lẽ bản thân họ không biết quan hệ giữa hai người, nhưng cách nhìn nhận của họ sẽ ảnh hưởng đến mong muốn hôm nay của họ, bạn phải có sức quan sát nhạy bén để tìm ra chân tướng. Phương pháp cũng rất đơn giản, trực tiếp hỏi một số vấn đề đơn giản, nhẹ nhàng có giá trị phán đoán. Đợi khi bạn có thể hiểu cách nhìn nhận của họ, làm thế nào để chuyển từ kỳ vọng về phát triển phần mềm của cá nhân sang chí hướng hiện tại (hay còn gọi là quá độ). Bạn thử nghĩ xem, mô hình chuyển đổi tâm lý này có đi ngược lại mục tiêu của nhóm? Có xung đột với giá trị

quan của nhóm? Chí hướng của họ có phù hợp với mục tiêu cuối cùng của nhóm.

Công việc của bạn là hướng dẫn họ đưa chí hướng của mình vào mục tiêu của nhóm, giúp anh ta biết cách phối hợp năng lực bản thân để đưa ra mục tiêu thích hợp.

Giao công việc thích hợp

Bây giờ, bạn đã biết họ muốn gì, và tại sao họ nghĩ như vậy; Sau đó tạm thời bỏ các thông tin này sang một bên, nghĩ xem họ thích hợp với vai trò gì, tìm ra ba việc mà họ có thể làm tốt hơn bất cứ người nào, nếu cho họ làm những việc này, họ có thể trở thành người thành công, yêu cầu các thành viên khác cho biết kỳ vọng đối với họ là gì, nếu họ khẳng định sở trường với ba việc đó, tìm cách chuyển chí hướng của họ thành kỳ vọng của mọi người.

Tối thiểu về lý luận, tăng cường sở trường của một người dễ hơn việc sửa khuyết điểm của anh ta. Nhưng là người lãnh đạo, bạn phải có dũng khí loại bỏ tất cả các giả thiết về phân công vai trò trong công việc, điều này nghe có vẻ dễ nhưng nhiều nhà quản lý đã thất bại nhiều lần. Tôi cho rằng chỉ có lãnh đạo xuất chúng mới làm được điều này: Điều chỉnh tổ chức, làm cho tài năng bẩm sinh của mỗi người đều được thể hiện hết sức.

BẦU TRỜI MỚI

Trong nhóm của tôi, có một chị có tư chất rất thông minh. Chị đã có 5 năm kinh nghiệm ở một công ty khác, đã làm phụ trách dự án rất thành công, do chúng tôi không hiểu trực tiếp kinh nghiệm trước đây của chị, do đó không biết phong cách và năng lực quản lý. Chị biểu thị muốn làm lãnh đạo cao hơn, cũng chính là, chị muốn làm công việc cả đời là "người quản lý".

Dần dần, chúng tôi nhìn ra năng lực không chê vào đâu của chị. Chị có thể giải quyết các vấn đề rắc rối trong giao tiếp, dù có xung đột và trở ngại thế nào đều có thể tìm ra phương pháp tốt nhất. Điều đó có nghĩa, chị có khả năng phân tích vấn đề, dẹp yên tranh chấp, càng ngày càng nhiều người nhờ chị hướng dẫn hay kiến nghị, hoặc mời chị phân tích các vấn đề phức tạp. Tất cả đều xảy ra rất tự nhiên, tôi không hề thúc đẩy.

Tôi quan sát và suy nghĩ kỹ hiện tượng này, đồng thời thảo luận với chị, tôi càng cảm thấy, một nhóm mạnh luôn cần người như chị, ngay lập tức có thể loại bỏ những nghi hoặc trong nhóm. Xem ra, có người chuyên giám thiểu ma sát, bỏ tranh chấp, cung cấp hiệu năng là một ý nghĩ rất hay, rất có ích cho nhóm. Nếu cho chị làm chủ nhiệm quản lý chắc là không thích hợp, chị không thể bộc lộ tài năng qua việc giám sát đốc thúc, chỉ cần mọi người ủng hộ chị là được rồi.

Tuy chúng tôi không nói về chi tiết nhưng chúng tôi đã có nhận thức chung, thành lập một nội dung công việc mới với chức danh mới để phù hợp với sở trường của chị, chức danh chưa quyết định nhưng nội dung công việc là "phân tích hiệu năng và cố vấn công việc". Chị vừa có thể tham gia nhóm tính năng với công việc là giải quyết các vấn đề hiệu năng trong công việc; vừa làm trưởng nhóm "phân tích hiệu năng và cố vấn công việc", phụ trách xây dựng lĩnh vực công việc mới này, phát triển quy tắc làm việc thích hợp, huấn luyện cấp dưới để bồi dưỡng các nhân tài tương tự cho công ty, tạo cơ sở phát triển cho vai trò mới này.

Tạo vị trí thích hợp cho nhân tài, có thể đem lại những lợi ích rất lớn cho tổ chức, đây là một kiểu "quản lý sáng tạo".

Khi bạn phát hiện khả năng đặc biệt của thành viên, cổ vũ họ phát huy, hầu hết mọi người đều có chút thái độ do dự. Tối thiểu tôi và mấy đồng nghiệp thân đều có tình huống này. Tôi nghĩ có thể do thiên phú là thứ quý giá của con người, trong thâm tâm vẫn luôn tự hào về điều đó, nhưng đối với bên ngoài lại có cảm giác cảnh giác. Chúng tôi cảm thấy do mình có thiên phú khác với mọi người nhưng lại rất sợ lấy ra công khai đánh giá, bởi sẽ có chút rủi ro - nhỡ đâu thất bại, không những mất thể diện mà cảm giác giá trị của bạn cũng bị đả kích.

TÔI CŨNG CẦN CỔ VŨ

Trước đây không lâu, cấp trên của tôi là Denis Gilbert đã giúp tôi khắc phục chướng ngại tâm lý này. Nhiều năm trước đây tôi hay diễn giảng nhiều nơi trên thế giới về chủ đề phát triển phần mềm, đánh giá của mọi người khá tốt, tôi cũng rất vui, không chỉ vì bản thân tôi được khẳng định, mà cũng là phản hồi của tôi đối với những người đã dạy dỗ mình (nội dung diễn giảng là phần trước cuốn sách này).

Denis biết tôi diễn thuyết thành công, rất nhiều nhân viên phát triển phần mềm rút ra nhiều lợi ích, nhưng đó là bên ngoài Microsoft, tôi không dám diễn thuyết trong công ty mình, kể ra những chuyện xấu và giằng xé nội tâm của mình trước mặt mọi người trong công ty thì thật xấu hổ, hơn nữa trong Microsoft tập trung rất nhiều cao thủ, nếu nói không hay, lại có tranh luận thì làm thế nào? Tôi tin rằng chia sẻ kinh nghiệm của mình rất có ích cho mọi người, nhưng tôi sợ bị công khai đánh giá, cho dù là vấn đề tôi tâm đắc thì cũng không dám chấp nhận rủi ro bị từ chối.

Denis không ngừng cổ vũ tôi, cho tôi niềm tin, diễn thuyết của tôi thực sự khá tốt, hơn nữa cuộc họp bàn kinh nghiệm cũng có ích cho rất nhiều người (Đương nhiên kể cả các đồng nghiệp trong Microsoft), giúp họ khắc phục khó khăn trong công việc (Bây giờ

Denis hối hận rồi, để tôi đi viết cuốn sách này, còn công việc của Visual C++ thì tạm thời bỏ qua). Sợ hãi và co mình lại thường cản trở sự phát triển của con người, đó là chướng ngại bên trong chứ không phải áp lực từ ngoại giới, trong môi trường được trao đủ quyền, tuy đây không phải là chướng ngại, do người quản lý tạo ra, nhưng người quản lý phải hỗ trợ cấp dưới phá bỏ chướng ngại này.

Mọi người rất nhạy cảm với thiên phú của mình, nếu cấp trên coi trọng, cho cơ hội và khuyến khích biểu hiện, anh ta sẽ làm tốt hơn so với tưởng tượng. Bạn phải luôn cổ vũ, hỗ trợ hành động mạo hiểm của anh ta, khi anh ta có biểu hiện tốt cần được khen ngợi đúng lúc. Lúc bắt đầu, anh ta sẽ đánh giá thấp khả năng của mình, cấp trên phải cho anh ta biết mình sẽ làm được, anh ta là thiên tài về lĩnh vực này, cấp trên tin rằng anh ta sẽ nhất định làm cho mọi người khâm phục, cấp trên có thể dùng các phương thức cho anh ta hiểu rằng không có gì quan trọng bằng thiên phú của anh, các khó khăn khác sẽ có thể được khắc phục.

Có rất nhiều ví dụ chứng minh, cấp trên tin tưởng và khẳng định cấp dưới. Khi được cấp trên khen ngợi tài năng, họ cảm thấy rất vui mừng và sẵn sàng nỗ lực, tuy nói "chết vì người tri kỷ" là hơi phóng đại nhưng họ sẽ phát huy tiềm lực lớn hơn; họ vốn làm công việc

thuộc sở trường của mình, bây giờ có biểu hiện càng tốt, trung thực hơn với công ty nhằm trả ơn.

Đối với một nhân viên chuyên nghiệp, có người hiểu và trọng dụng anh ta, thì cũng đáng để báo đáp.

Cấp trên có tâm sẽ biết tìm ra phương pháp tốt để hoà hợp với thế giới nội tâm của cấp dưới, đây là một thách thức rất khó nhưng rất đáng làm. Cần bỏ thời gian đi tìm hiểu cấp dưới, có lẽ nửa tiếng nói chuyện cũng mất 2 giờ chuẩn bị, suy nghĩ xem bạn sẽ nói những thông tin gì, thông tin cần được truyền đạt rõ ràng, không gây hiểu lầm hay làm anh ta suy nghĩ lung tung.

Không nên nói quá nhiều trong một lần, nhưng thông tin mỗi lần đều rất rõ ràng, vận dụng khả năng hoà hợp của bạn, đó chính là nhiệm vụ quan trọng của cấp trên, hoà hợp rõ ràng là điều mà cả hai bên đều cần.

Bạn phải "tái xác định" thông tin của mình, động tác này phải để cấp dưới biết, bạn không nói tuỳ hứng, bạn thực sự khẳng định thiên phú của anh ta, bây giờ bạn trao quyền cho anh ta.

Khiêm tốn học hỏi

Khi bạn gặp vấn đề, cố gắng xin chỉ giáo của mọi người xung quanh, tuy bạn đã khá giỏi, nhưng vẫn có

người giỏi hơn bạn trong một lĩnh vực nào đó, có khả năng giúp bạn. Do đó, không nên kiêu ngạo, mở miệng xin giúp đỡ mới là biện pháp hay. Bạn không phải là người đơn độc, xung quanh bạn có rất nhiều nhân tài có thể giúp bạn. Không nên sợ rằng hỏi người khác sẽ làm lộ yếu điểm của bạn, ngược lại, cho mọi người thấy yếu điểm của bạn càng làm cho họ chấp nhận bạn, cấp dưới càng trung thực với bạn.

Xây dựng mục tiêu ngắn hạn

Cho thành viên mục tiêu ngắn hạn để họ thể hiện tài năng, chứng minh tiến bộ của họ, cho họ biết bạn hỗ trợ và tạo cơ hội tốt nhất, điều này cũng có tính tương hỗ, bạn cũng mong muốn họ thể hiện sự ưu việt, nếu họ biểu thị không cần thì tạm thời không nói làm gì; bạn có thể cho họ biết đây là cơ hội rất tốt, trong cả nhóm, bạn tin họ thích hợp nhất để đảm nhiệm công việc này, và bạn sẵn lòng tạo điều kiện tốt nhất, nhưng với điều kiện là cấp dưới tự nguyện làm mới được. Bây giờ cho họ lựa chọn, nếu họ không thích bạn có thể chuyển cơ hội cho người khác.

Xây dựng kế hoạch dài hạn

Sau khi giao nhiệm vụ mới cho một thành viên, anh ta sẽ hy vọng có một khái niệm khá dài về nhiệm vụ này, hoặc là hướng phát triển lâu dài, ... anh ta cũng muốn biết mình phải nắm cái gì trên nguyên tắc nào.

Có lẽ anh ta ngại hỏi bạn, nhưng đây là công việc quan trọng của người quản lý, chớ có lười!

Xây dựng chế độ đánh giá

Làm sao để bạn biết được thành viên trong nhóm tốt hay xấu. Nhóm không thành công thì thất bại, làm thế nào để biết mức độ cống hiến (hay phá hoại) của một thành viên?

Công việc phát triển phần mềm cũng như trò chơi điện tử vậy, có rất nhiều mức độ, độ khó ngày càng lớn. Mỗi thành viên đều mong được thành tựu ở mỗi mức độ, mỗi mức độ đều có thách thức, lúc mới bắt đầu mọi người đều bị mê hoặc, cảm thấy rất khó khăn, sau đó bắt đầu hưng phấn thậm chí phát cuồng. Rất khó khăn mới đến mức độ sau, quy tắc chơi lại khác, phương thức hợp tác trước chưa chắc có tác dụng, làm không được lại thành dở. Hết mức này đến mức khác, nhóm cuối cùng học được cách đối phó với môi trường mới, đồng thời phát triển thành phương thức hợp tác thực sự hữu hiệu.

Đánh giá biểu hiện nhóm, quan trọng nhất là quyết định xem họ đang ở mức độ nào, dễ hay khó (số người nhóm, trình độ biểu hiện của mỗi người, tính chất nhiệm vụ dự kiến không. Đây là nguyên tắc cơ bản trong đánh giá thành tích nhân viên.

Nhóm phát triển phần mềm cũng như trò chơi

điện tử, xem bạn có bao nhiêu "đạn", "số lần", điều kiện tiên thiên càng tốt càng dễ thắng, cấp trên phải chú ý đến điểm này.

Đánh giá, là lúc bạn tái kiểm tra kỳ vọng của cấp dưới, xem xem bạn có bỏ qua thiên phú của ai không? có ai muốn điều chỉnh bước đi? mục tiêu ngắn hạn có quá cao hay quá thấp. Hoà hợp trực tiếp và thẳng thắn là cách làm tốt nhất, tuy cũng là cách làm khó nhất, thành viên cũng cần có cơ hội này để cùng bạn xem xét, thách thức của mình có cần điều chỉnh không? Mức độ này có cần làm lại? hay là nhảy đến mức độ này? tốc độ thông thường quá nhanh mà cũng không tốt lắm, có lẽ cần luyện tập ở mức độ này, hay là vẫn chưa học hết những thứ ở mức trước.

Quá trình thành thạo của mỗi người trong từng mức độ đều khác nhau, có người thành thạo nhanh chóng, có người mãi vẫn không quen. Mỗi người đều có đường đi riêng, trải qua lịch trình thạo riêng và đưa ra sắc thái riêng cho mình. Về cơ bản, ở mức độ nào cũng có hợp tác nhóm, mọi người hỗ trợ lẫn nhau mới có thể lên mức độ cao hơn, sau đó nhìn rộng hơn, xa hơn, sức ảnh hưởng của từng người càng lớn, vai trò càng phức tạp. Sau cùng, bồi dưỡng nên khả năng lãnh đạo xuất chúng (không phải khả năng giám sát), hoặc hiệu năng công việc cao hơn, cuối cùng trở thành tài sản quý giá nhất của công ty.

NGUỒN TƯ LIỆU

Rất khó có thể nói chính xác cách nghĩ của sách này bắt nguồn từ đâu. Về cơ bản, nội dung cuốn sách chủ yếu là những tâm đắc của tôi sau nhiều năm làm việc, những cuốn sách tôi đã đọc cho tôi một số gợi ý, nhiều người đóng góp cho tôi (tôi đã nói đến trong phần giới thiệu, nay cảm ơn họ lần nữa). Để bạn đọc tiện tìm kiếm các nguồn tư tưởng, tôi liệt kê qua một số sách và người mà tôi nhớ đến, hy vọng sẽ giúp ích bạn đọc.

Chắc hầu như ai cũng biết về Freud. Đọc các trước tác của ông, hoặc truyện ký, phim hay bất cứ phương thức nào để hiểu ông, bạn sẽ thấy một số huyền bí về tâm lý học. Nếu có thời gian, hãy làm thử nghiệm phân tích tâm lý học, giá thành tuy cao nhưng có thể bạn sẽ nhận được kiến thức sâu sắc về tâm lý học, có cách nhìn mới mẻ hơn.

Darwin là nhà thám hiểm tự nhiên vĩ đại, lý luận của ông có thể dùng trong hầu hết các lĩnh vực. Bỏ chút thời gian nghiên cứu tác phẩm của ông, nhất là cuốn "The Selfish Gene" (Gien ích kỷ) bạn sẽ thấy những khởi phát rất đánh giá, và hiểu rõ hơn về bản chất biến hoá của xã hội cạnh tranh này.

Thưởng thức tác phẩm của Shakespeare là một chủ ý tốt, không cần mất nhiều thời gian đọc hết toàn tập, quan trọng nhất là bạn nắm được yếu lĩnh thẩm

mỹ, bạn sẽ thấy sự huyền diệu của lãnh đạo và cổ vũ, chỉ như vậy cũng đủ để tác phẩm của ông trở thành kinh điển.

Tổng thống Lincoln, trước General Grant và Churchill luôn hấp dẫn tôi. Tướng Grant toàn tâm ý, với tính cách không ngừng chiến đấu cho đến khi trận chiến kết thúc, chính là nhà lãnh đạo nhóm phát triển phần mềm xuất sắc. Churchill cũng là người có tính cách mạnh mẽ, khi cả thế giới phản đối ông vẫn kiên trì lý tưởng của mình, ông giỏi diễn thuyết, có thể khích lệ ý chí chiến đấu bất khuất. Tổng Lincoln cũng là nhà diễn thuyết nhiệt tình, có thể khích lệ những người ích kỷ nhất, là thiên tài trong lãnh đạo.

Phần mềm là sản vật tinh tế hiện đại của trí tuệ con người, do đó bạn phải bồi dưỡng khả năng quan sát, thưởng thức bức vẽ của Picasso với phương thức biểu hiện phá bỏ truyền thống, vượt cả hiện thực, có lẽ sẽ cho bạn một chút linh cảm đặc biệt. Các đồ vật thẩm mỹ khác như quần áo thịnh hành, kiến trúc, nghệ thuật, thiết kế đều có bối cảnh văn hoá thời đại, nói lên giá trị quan và trào lưu tư tưởng của mọi người. Do đó chỉ cần bạn có thời gian, không nên tiếc học thưởng thức cái đẹp.

Về thẩm mỹ, trong một chương tôi có nhắc qua. Tôi đề nghị bạn đọc tác phẩm của Rudilf Arnheim, nhất

là "Cảm nhận nghệ thuật và thị giác" (Art and Visual Perception), bạn sẽ tìm được mọi lý luận khi thiết kế giao diện người dùng, nó sẽ giúp bạn thiết kế các giao diện xuất sắc.

Có một cuốn sách rất hay có thể tăng quan niệm lịch sử cơ bản cho bạn, được Will và Visual Durant cùng viết trong "Câu chuyện văn minh" (The Story of Civilization). Nếu bạn muốn tạo ra sản phẩm vĩ đại, lịch sử là bài học bắt buộc, bạn cần phải hiểu lịch trình văn minh nhân loại, nếu không sẽ không có duyên với các phần mềm lớn.

Bộ phim "Bữa tiệc Babette" (Babette's Feast) là bài học nhập môn của các nhà lãnh đạo, bạn có thể hiểu về nhận thức của mọi người, hầu hết thời gian của người lãnh đạo nên giành cho phương thức để mọi người dùng nhiều hơn và trải qua nhiều sự việc phong phú. Bộ phim này rất hay.

Còn sứ mệnh và nhiệm vụ của nhà nghệ thuật, tôi đề nghị bạn xem bộ phim "Ed Wood".

Nếu bạn không muốn tìm nhiều nguồn cảm hứng nữa, có thể tiếp tục vùi đầu vào làm phần mềm, nhưng nếu không có những nguồn mới rót vào thì đừng mong làm ra phần mềm hay.

MỤC LỤC

Lời nhà xuất bản 5
Lời giới thiệu 7
Giới thiệu tác giả 11
Lời tác giả 13

KHÁI QUÁT. 4 THỜI KỲ PHÁT TRIỂN PHẦN MỀM 17
Ý tưởng phát triển phần mềm 20
4 thời kỳ phát triển phần mềm 25

PHẦN 1. GIAI ĐOẠN BỐ CỤC 28
Nguyên tắc 1. Xây dựng mục tiêu chung 36
Nguyên tắc 2. Làm cho mọi người chủ động tham gia 52
Nguyên tắc 3. Xây dựng kế hoạch kỹ thuật phát triển nhiều phiên bản 56
Nguyên tắc 4. Hãy động não 67
Nguyên tắc 5. Thăm dò đối phương 75
Nguyên tắc 6. Chú ý tỷ lệ nhân viên 82
Nguyên tắc 7. Sử dụng nhóm giám sát tính năng 84
Nguyên tắc 8. Trách nhiệm của quản trị dự án 102
Nguyên tắc 9. Uy quyền chứ không phải bạo quyền.... 109
Nguyên tắc 10. Không có đối thủ chưa chắc đã là điều tốt 121
Nguyên tắc 11. Đối thủ đuổi sát? Tung ra những tính năng mới 125
Nguyên tắc 12. Tụt hậu? Tăng cường đầu tư, nhanh chóng tung ra phiên bản mới 128
Nguyên tắc 13. Dẫn đầu? Không được quay lại 132
Nguyên tắc 14. Đảm bảo sự mới mẻ 134
Nguyên tắc 15. Làm khách hàng vui mừng 144
Nguyên tắc 16. Tìm kiếm hồng tâm 148

Nguyên tắc 17. Xây dựng quan hệ với khách hàng, chứ không đơn thuần là bán sản phẩm 151
Nguyên tắc 18. Tăng chu kỳ sản phẩm 156
Nguyên tắc 19. Tìm kiếm sự vượt trội..... 162
Nguyên tắc 20. Thiết lập chủ đề 163
Nguyên tắc 21. Không dựa vào những việc không xác định 170
Nguyên tắc 22. Giảm sự phẫn nộ của khách hàng 171
Nguyên tắc 23. Khả năng cấy ghép của phần mềm 172
Nguyên tắc 24. Khi thiết kế cần suy nghĩ đến nhân tố thời gian 173
Nguyên tắc 25. Từ chối mệnh lệnh không hợp lý 178
Nguyên tắc 26. Coi công việc như trò chơi 182

PHẦN 2. GIAI ĐOẠN GIỮA 184
Nguyên tắc 27. Phương pháp bác sỹ..... 187
Nguyên tắc 28. Tam giác vàng phát triển phần mềm... 189
Nguyên tắc 29. Không giả vờ hiểu..... 191
Nguyên tắc 30. Xây dựng điểm kiểm tra thích hợp..... 196
Nguyên tắc 31. Cẩn thận không có thành viên ở điểm kiểm tra 202
Nguyên tắc 32. Thường xuyên tích hợp, phần mềm sẽ thuận lợi ra đời 207
Nguyên tắc 33. Nắm vững tình hình thực tế 214
Nguyên tắc 34. Cột mốc không hụt 219
Nguyên tắc 35. Mọi thành viên đều đạt tới cột mốc không hụt..... 224
Nguyên tắc 36. Sau mỗi cột mốc, cần bình tĩnh kiểm điểm 225
Nguyên tắc 37. Nắm vững ý nghĩa và tinh thần thực chất của cột mốc..... 227
Nguyên tắc 38. Bồi dưỡng nhóm bình thường..... 230

Nguyên tắc 39. Quá nhiều mốc sẽ không nắm vững ... 237
Nguyên tắc 40. Mỗi cột mốc cần có tôn chỉ riêng 238
Nguyên tắc 41. Tìm kiếm cột mốc xuất hiện tự nhiên .. 242
Nguyên tắc 42. Nếu trượt, hãy đứng lên 252
Nguyên tắc 43. Không sửa kỳ hạn cuối cùng chỉ vì tiến độ tụt hậu 274
Nguyên tắc 44. Lỡ cột mốc này, nhưng phải đạt được cột mốc sau 278
Nguyên tắc 45. Coi tụt hậu là cơ hội học tập 280
Nguyên tắc 46. Nhìn cây thấy rừng 282
Nguyên tắc 47. Thế giới thay đổi, bạn cũng phải thay đổi theo 283

PHẦN 3. GIAI ĐOẠN HOÀN THÀNH 287

Nguyên tắc 48. Quan tâm nhiều hơn yêu cầu..... 297
Nguyên tắc 49. Bản thử nghiệm beta không phải là lúc sửa đổi chức năng 300
Nguyên tắc 50. Bản beta là động tác khởi động 301
Nguyên tắc 51. Thuật cấp cứu..... 303
Nguyên tắc 52. Đảm bảo tính ổn định của phần mềm . 306

PHẦN 4. GIAI ĐOẠN CÔNG BỐ 307

Nguyên tắc 53. Phần mềm lớn phải có câu chuyện lớn 313
Nguyên tắc 54. Xây dựng hình tượng chiến thắng 321
Lời kết 323

PHỤ LỤC. GIỎI DÙNG NGƯỜI..... 324

Thuê nhân viên thông minh..... 326
Phân việc đúng người..... 331
Nguồn tư liệu..... 347

NHÀ XUẤT BẢN VĂN HOÁ - THÔNG TIN
43 Lò Đúc - Hà Nội

CHÌA KHOÁ TRÊN ĐƯỜNG HỘP NHẬP

BÍ QUYẾT THÀNH CÔNG CỦA TẬP ĐOÀN MICROSOFT

Người dịch: Mạnh Linh - Minh Đức

Chịu trách nhiệm xuất bản:
VŨ AN CHƯƠNG
Chịu trách nhiệm bản thảo:
PHẠM NGỌC LUẬT

Biên tập: **THANH VIỆT**
Sửa bản in: **PHÒNG CHẾ BẢN**
Vẽ bìa: **TRẦN ĐẠI THẮNG**

PHÁT HÀNH TẠI: NHÀ SÁCH MINH LÂM
18 Ngô Quyền, Hà Nội - ĐT: (04) 934.4435

In 1.000 cuốn, khổ 13 x 19 cm, tại Công ty In Khoa học Kỹ thuật 101A Nguyễn Khuyến - Hà Nội. Số in: 720. Giấy phép xuất bản số: 449-XB-QLXB/63-VHTT. In xong và nộp lưu chiểu quý III/2003.

BÍ QUYẾT THÀNH CÔNG
CỦA TẬP ĐOÀN MICROSOFT

Chìa khóa
trên đường hội nhập

Với kinh nghiệm phong phú của mình, tác giả dựa vào kết tinh suy nghĩ và những bài học đau thương để tổng kết thành 54 nguyên tắc đơn giản. Jim Mc Carthy phân tích các chiến lược thích hợp nhất trong các tình huống cạnh tranh, hướng dẫn bạn cách cứu vãn tình thế trong trường hợp chẳng may tụt hậu. Ngoài ra bạn còn học được cách vận dụng những bí quyết của các nhân tài...

Cũng với kinh nghiệm sâu sắc và văn hoá tinh thần độc đáo của Jim Mc Carthy. Bạn sẽ thấy cuốn sách thú vị cuốn hút như mình đang phát triển phần mềm vậy.

Cuốn sách viết cho lập trình viên, chuyên viên phát triển, nhân viên kinh doanh quản lý kỹ thuật, cũng như những người muốn tìm hiểu những bí mật trong phát triển phần mềm, hy vọng rằng các bạn sẽ thấy những điều bổ ích từ cuốn sách.

GIÁ: 35.000Đ